DOCUMENT RESUME

ED 343 560 IR 015 437

AUTHOR	Birnbaum, Lawrence, Ed.
TITLE	Proceedings of the International Conference on the Learning Sciences (5th, Evanston, Illinois, August 1991).
INSTITUTION	Association for the Advancement of Computing in Education, Charlottesville, VA.
REPORT NO	ISBN-1-880094-01-0
PUB DATE	91
NOTE	458p.
PUB TYPE	Collected Works - Conference Proceedings (021)
EDRS PRICE	MF01/PC19 Plus Postage.
DESCRIPTORS	*Artificial Intelligence; Authoring Aids (Programing); Cognitive Processes; Computer Simulation; Computer Software; Cooperative Learning; Elementary Secondary Education; Higher Education; Hypermedia; Industrial Training; *Learning Strategies; Lesson Plans; Media Research; Programed Tutoring; Psychological Studies; Science Education; Writing Skills

ABSTRACT

This volume contains the papers presented at the fifth International Conference on the Learning Sciences (formerly the International Conference on Artificial Intelligence and Education). As the change in name signifies, a strong effort has been made to open the conference to a broader audience, including educational and cognitive psychologists interested in the problems of teaching and learning, and software designers interested in building systems to facilitate teaching and learning. The issues involved in artificial intelligence (AI) and education are of interest to a broad spectrum of researchers in education, psychology, and software systems. As in previous years, there are papers on tools and authoring systems, teaching architectures, student modelling, interfaces, simulation, and empirical evaluation. Science education, especially biology, mathematics, and physics, receives attention, as do language and writing skills. Problems of corporate and industrial training are also addressed. Cooperative and collaborative learning involving groups of students is an important theme of many papers. Questioning and story-telling are taken as central to teaching and learning, as are the promotion of development and use of cognitive strategies. A number of papers address the question of how computer systems can aid teachers in such tasks as lesson planning. Finally, interactive video and hypermedia systems continue to grow in importance. Each of the 61 papers includes its own abstract and references; only the abstract is provided for six of the invited keynote papers. (DB)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

The International Conference on the Learning Sciences

Proceedings of the 1991 Conference

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☐ This document has been reproduced as received from the person or organization originating it
☐ Minor changes have been made to improve reproduction quality

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

ED343560

The Institute for the
Northwestern University
Evanston, Illinois, USA
August 1991

Editor: Lawrence Bir

IR015437

"PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY

Gary H. Marks

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)."

BEST COPY AVAILABLE

Published by
Association for the Advancement of Computing in Education

The International Conference on the Learning Sciences

Proceedings of the 1991 Conference

The Institute for the Learning Sciences
Northwestern University
Evanston, Illinois, USA

Lawrence Birnbaum, Editor

Association for the Advancement of Computing in Education
Charlottesville, Virginia, USA

Program Committee
Gregg Collins
Andrea A. diSessa
Richard Feifer
Gary Marks
David Neves
Stellan Ohlsson
Roy Pea
Elliot Soloway

Sponsored in part by
Association for the Advancement of Computing in Education, publisher of the *Journal of Artificial Intelligence in Education*

ISBN 1-880094-01-0

Copyright © 1991 by Association for the Advancement of Computing in Education

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without permission in writing from the publisher.

The publisher is not responsible for the use which might be made of the information contained in this book.

Published by Association for the Advancement of Computing in Education
P.O. Box 2966, Charlottesville, Virginia 22902, USA

Printed in the USA

Preface

This volume contains the papers presented at the fifth International Conference on the Learning Sciences (formerly the International Conference on Artificial Intelligence and Education), held at Northwestern University, Evanston, Illinois, USA, in August 1991. As the change in name signifies, we have made a strong effort to open the conference to broader participation this year. The issues involved in AI and education are of interest to a broad spectrum of researchers in education, psychology, and software systems, and our success in addressing these issues will ultimately require an equally diverse base of knowledge and experience. In this spirit we have encouraged greater participation by, on the one hand, educational and cognitive psychologists interested in problems of teaching and learning, and on the other, software designers interested in building systems to facilitate teaching and learning–whether or not they draw their inspiration from AI. As a result, this volume includes papers on topics ranging from conceptual change and analogical learning through hypermedia systems and organizational memory.

In between, there is plenty of "AI and education" as well. As in previous years, there are papers on tools and authoring systems, teaching architectures, student modelling, interfaces, simulation, and empirical evaluation. Science education, especially biology, mathematics, and physics, continues to receive a great deal of attention, as do language and writing skills. Problems of corporate and industrial training are also addressed.

A number of other trends are also apparent. Cooperative and collaborative learning involving groups of students is an important theme in many papers. Questioning and storytelling are taken as central to teaching and learning by much of the work reported here. Promoting the development and use of appropriate representations and cognitive strategies by students also receives a great deal of attention. A number of papers address the question of how computer systems can aid teachers in such tasks as lesson planning. Finally, interactive video and hypermedia systems continue to grow in importance.

The invited speakers–John Bransford and Vanderbilt University's Cognition and Technology Group, Allan Collins (Northwestern University and Bolt Beranek and Newman, Inc.), Andrea diSessa (University of California, Berkeley), James Greeno (Stanford University), Alan Kay (Apple Computer, Inc.), Marlene Scardamalia (Ontario Institute for Studies in Education), Roger Schank (Northwestern University), and Elliot Soloway (University of Michigan)–skillfully represented a wide variety of viewpoints and disciplines.

All of these trends indicate healthy growth in new and interesting directions. Above all, there seems to be a growing desire to field real systems (or at least, serious prototypes) that can help make a difference in overcoming the difficult and important educational challenges our society now faces.

Finally, the conference organizers wish to express their deep gratitude to the Association for the Advancement of Computing in Education for their sponsorship and for publishing these proceedings.

Lawrence Birnbaum

Contents

ANDERSON & PELLETIER
A Development System for Model-Tracing Tutors 1
BAKER
The Influence of Dialogue Processes on Students' Collaborative Generation of Explanations for Simple Physical Phenomena 9
BENTO
The Evaluation of Innovative Computer-Based Learning Systems: Teaching Newtonian Mechanics 20
BERTIN, BUCIOL, DONDOSSOLA & CRISTINA LANZA
An Adaptive ITS Architecture for an Industrial Training Problem 32
BIELACZYC & RECKER
Learning to Learn: The Implications of Strategy Instruction in Computer Programming 39
BLEVIS & KASS
Teaching by Means of Social Simulation 45
BREDEWEG & WINKELS
Teaching According to GARP 52
BREUKER
Learning Construction Problem Solving 59
BREWER & CHINN
Entrenched Beliefs, Inconsistent Information, and Knowledge Change 67
BRUNEAU, CHAMBREUIL, CHAMBREUIL, CHANIER, DULIN, LOTIN & NEHEMIE
Cognitive Science, Artificial Intelligence, New Technologies: How to Cooperate for a Computer-Assisted Learning to Read System 74
BURNS, GRAY & RADLINSKI
Tuning the Ideal Student Model: Towards Intelligent Editing of ITS Models 83
CARROLL, SINGLEY & ROSSON
Toward an Architecture for Instructional Evaluation 85
THE COGNITION AND TECHNOLOGY GROUP AT VANDERBILT (Keynote Lecture)
Designing Environments that Invite Thinking and Innovation 91
COLLER, PIZZINI, WOGULIS, MUNRO & TOWNE
Direct Manipulation Authoring of Instruction in a Model-Based Graphical Environment 92
COLLINS (Keynote Lecture)
Design Issues for Interactive Learning Environments 100
COOPER
An Architecture for Apprenticeship: Collaboration with an Intelligent Tutoring System for Qualitative Electrical Troubleshooting 101
CORBETT, ANDERSON & FINCHAM
Menu Selection vs. Typing: Effects on Learning in an Intelligent Programming Tutor 107
DEDE
Designing a Tool for Imaging Mental Models Underlying Training 113

DISESSA (Keynote Lecture)
Computational Media as a Foundation for New Learning Cultures 121
DU & MCCALLA
CBMIP—A Case-Based Mathematics Instructional Planner 122
DUNCAN, HUNT, GLUSHKO & ROUSE
The Space Shuttle Fuel Cell Tutor: A Hybrid Training Environment 130
EDELSON
Why Do Cheetahs Run Fast?
Responsive Questioning in a Cased-Based Teaching System 138
ESKRIDGE
Continuous Analogical Learning 145
FEIFER & SOCLOF
Knowledge-Based Tutoring Systems:
Changing the Focus from Learner Modeling to Teaching 151
FERGUSON, BAREISS, OSGOOD & BIRNBAUM
ASK Systems: An Approach to Story-Based Teaching 158
FISCHER
Supporting Learning on Demand with Design Environments 165
FORBUS
Towards Tutor Compilers:
Self-Explanatory Simulations as an Enabling Technology 173
FREDERIKSEN, DONIN, DECARY & EMOND
Discourse-Based Second-Language Learning Environments 180
FRYE
Cognitive Strategies, Learning, and Educational Software 187
GALDES, SMITH, SMITH, JR.
Determining When to Interrupt and What to Say:
An Empirical Study of the Case-Method Tutoring Approach 194
GOODYEAR, JOHNSON, VAN KOUWEN & RONTELTAP
Uses of Interactivity in Tutorial Support for Exploratory Learning 203
GRAESSER, LANGSTON & LANG
Designing Educational Software Around Questioning 204
GREENO (Keynote Lecture)
Environments for Situated Conceptual Learning 211
HALL & OHLSSON
A Theory of Manipulatives in Arithmetic Learning 217
HARTLEY, BYARD & MALLEN
Qualitative Modeling and Conceptual Change in Science Students 222
HINTZE
Guided Discovery Learning:
The Computer as a Cooperative Learning Assistant 231
JACOBSON & SPIRO
Hypertext Learning Environments and Cognitive Flexibility:
Characteristics Promoting the Transfer of Complex Knowledge 240
JONA, BELL & BIRNBAUM
Using Button Theory to Structure Student-Teacher Interaction in
Computer-Based Learning Environments 249

KASS & GURALNICK
Environments for Incidental Learning: Taking Road Trips Instead of Memorizing State Capitals 258
KAY (Keynote Lecture)
Point of View Is Worth 80 IQ Points 265
KOEDINGER
On the Design of Novel Notations and Actions to Facilitate Thinking and Learning 266
KOLODNER
Helping Teachers Teach Science Better: Case-Based Decision Aiding for Science Education 274
KRUPER & NELSON
Developing and Evaluating Novel Strategic Computer Simulations: A Case Study in Translating an Instructional Pedagogy to the Classroom 281
LAFFEY, MACHIRAJU & CHANDHOK
Organizational Memory as a Support for Learning and Performance: Prototypes and Issues 289
LESTER & PORTER
A Student-Sensitive Discourse Generator for Intelligent Tutoring Systems 298
LEWIS & NATHAN
A Framework for Improving Students' Comprehension of Word Arithmetic and Word Algebra Problems 305
MCARTHUR & LEWIS
Overview of Object-Oriented Microworlds for Learning Mathematics through Inquiry 315
MOHAN, GREER & JONES
Adapting Instruction to the Cognitive Needs of Learners in an Intelligent Tutoring System 322
MONARCH, BOYLE & DYKSTRA, JR.
Conceptual Restructuring in History of Mechanics as a Model for Learning Mechanics 329
NEVES, CAMPBELL, GREISING & GREENLEE
Cascaded Problem Sets: A Model for Learner-Driven Instruction 338
NEWMAN
Computer Support for School Work 344
OHLSSON & BEE
Radical Strategy Variability: A Challenge to Models of Procedural Learning 351
PALTHEPU, GREER & MCCALLA
Learning by Teaching 357
PLOGER
Learning Biology Via Programming 364
RAGHAVAN, SCHAUBLE & GLASER
A Graphic Notation to Support Reflection in Scientific Reasoning 370
REIMANN & SCHULT
Modeling Example Elaboration Strategies 375
REISER, BEEKELAAR, TYLE & MERRILL
GIL: Scaffolding Learning to Program with Reasoning-Congruent Representations 382

RESNICK
Beyond the Centralized Mindset 389
SCARDAMALIA (Keynote Lecture)
An Architecture for the Social Construction of Knowledge 397
SINGLEY, ANDERSON & GEVINS
Promoting Abstract Strategies in Algebra Word Problem Solving 398
SOLOWAY (Keynote Lecture)
"Fermat's Last Theorem? I learned about it on Star Trek" 405
TWIDALE
Cognitive Agoraphobia and Dilettantism: Issues for Reactive Learning Environments 406
VASANDANI & GOVINDARAJ
An Experimental Evaluation of an Intelligent Tutor for Diagnostic Problem Solving 414
WALLACE, SILBERSTEIN & BLUFF
How Many Modes of Cognitive Operation? Some Evidence from a Brain Monitoring Study of Quantification Processes 422
WHITELOCK, TAYLOR, O'SHEA, SCANLON, CLARK & O'MALLEY
How Students Construct a Shared Understanding of Collisions in Newtonian Mechanics 430
WINKELS, BREUKER & DEN HAAN
Principles and Practice of Knowledge Representation in EUROHELP 442

A Development System for Model-Tracing Tutors

John R. Anderson and Ray Pelletier
Department of Psychology
Carnegie Mellon University, Pittsburgh, PA 15213 USA

Abstract: Our model-tracing methodology for tutoring is based on the ACT* theory of skill acquisition. According to that theory a skill can be analyzed into a set of productions rules and instruction can be organized around these rules. Students' problem-solving behavior can be interpreted and tutored by tracing their solution through the production rules. Problems can be selected to optimize mastery of production rules. Because of recent advances in production-system efficiency we have been able to develop a general system for building model-tracing tutors. Developing a tutor in this system involves constructing an interface, defining the production rules, creating declarative instruction, and setting up the knowledge tracer to manage the curriculum.

Since 1984 we have developed a number of tutoring systems which have all exemplified what we have called the model-tracing approach to intelligent tutors. The first three systems were a tutor for the programming language LISP (Anderson & Reiser, 1985), a tutor for proofs in high school geometry (Anderson, Boyle, & Yost, 1985), and a tutor for symbol manipulation in high school algebra (Lewis, Milson, & Anderson, 1987). All three systems have been used in actual classroom situations with generally positive outcomes. Some evaluations of these systems are reported in Anderson, Boyle, Corbett, and Lewis (1990). More recently we have developed and deployed at CMU a single tutoring architecture which teaches LISP, Pascal, and Prolog to undergraduate freshman (Anderson, Corbett, Fincham, Hoffman, & Pelletier, in press) with generally positive results. We have also developed a tutor for high school word algebra problems which has had successful laboratory tests but has not yet been deployed in the classroom (Singley, Anderson, Givens, & Hoffman, 1989). Although specific evaluations produce varying magnitudes of effects our general estimation is that our tutors improved student achievement by about one standard deviation given constant amount of time or at least halved the time to reach the comparable levels of achievement.

Each of these tutors has had a production system model of the skills they were supposed to teach. This involved developing a set of rules which in combination would perform the target skill. A typical tutor would involve as many as 500 of these rules. These rules reflected a cognitive model which we thought could underlie successful problem solving in that domain. Thus, for instance, we would have a production rule for geometry of the following form:

IF the goal is to prove $\triangle ABC$ is congruent to $\triangle DBE$
    and AE and CD are collinear
THEN infer $\angle ABC$ is congruent to $\angle DBE$ because they are
    vertical angles.

The rule reflects the wisdom that it is useful to make a vertical angle inference when trying to prove triangles congruent.

Organizing a set of production rules which will solve a class of problems is like creating an expert system with the added constraint that it be cognitively plausible. This is not a particularly easy task but having done it we are in a powerful position to deliver instruction. The cognitive model defines the instructional objectives–basically to communicate to the student each of the underlying production rules. Student interaction with our tutors takes the form of solving problems while interacting with the computer system. The tutors could use their production system models to interpret the students' behavior and provide feedback. Basically, our systems try to find some sequence of production rules which will match the students' behavior and use this sequence as an interpretation of the students' problem solving. The approach is called model tracing because it traces the students' behavior with a set of production rules.

Each of the tutors we built was more or less constructed anew. There was only informal transfer from our experience in constructing previous tutors. We felt that we had reached the point where we could articulate what we were doing when we built a computer tutor and create a system that facilitated that process. We thought it would be useful to codify this as a development system for a number of reasons. First, it would provide a precise definition of what our approach is to tutor construction. Second, it could be made available to others who wanted to create model-tracing tutors. Third, we felt we could endow such a system with facilities which would improve the process of developing such a tutor and the performance of the tutor once developed.

We will describe in this paper the system we have developed for building model-tracing tutors (Anderson et al., in press). It has evolved as a generalization of the multiple programming languages tutor. Its generality is currently being tested by having it embody the word problem tutor (Singley et al., 1989). It is best described as a development system and not an authoring system. It does not eliminate the need to go through the same steps as we did when we created our first tutor. Rather it just structures and facilitates the development process.

In this paper we will go through the basic principles which we feel lie behind our approach to tutor development. Then we will describe the essential steps involved in creating a tutor in our system.

Principles of Model-Tracing Tutors

We have previously (Anderson, Boyle, Farrell, & Reiser, 1987) published a list of cognitive principles for the development of tutors based on the ACT* theory (Anderson, 1983) of cognition. We have subsequently realized that the list was overly elaborate and involved issues not really central to the conception of skill acquisition in ACT*. The basic implications of ACT* for instruction are really quite simple but important.

The first principle derived from ACT* is that it is essential to define the target cognitive model as a set of productions rules. Without this one does not have a well-defined educational goal within our approach. The task of building such a model is not easy although we have developed a number of helpful facilities within our system. Our view is that this is the most labor intensive aspect of tutor development. It is also worth noting that there is not necessarily a unique set of production rules which prescribe how students should approach a task. Just as there are multiple possible programs for doing a task there are multiple possible production systems. There is no guarantee that one will come up with the best production system for performing a task. Some of our recent work on geometry tutoring is predicated on the belief that we have found a more powerful cognitive model for proof construction (Koedinger & Anderson, 1990) than the tutor which underlay the original geometry tutor.

The second principle concerns how these productions are to be communicated to the student. The ACT* theory has a quite specific proposal for how production rules are acquired. According to ACT* one cannot directly tell students the underlying rules. Rather, students form these rules as a byproduct of problem-solving. Their initial attempts to solve a problem involving a rule take the form of analogy to some example that illustrates the rule. Thus, one needs to communicate the rules to students by providing them with examples that illustrate the rules. The examples need to be embellished to communicate the underlying rules. Thus, to communicate the production rule given earlier one might present the student with an example and the accompanying embellishment "When trying to prove triangles congruent that form a vertical angle configuration it is a good idea to infer that the vertical angles are congruent." When the student next comes across a problem that involves this rule, they can solve that problem by analogy to this example. If they do, the production rule will be formed by the knowledge compilation process in ACT* (Anderson & Thompson, 1989).

The third principle of tutoring that derives from ACT* is that one wants to maximize the rate at which students have opportunities to form and practice these production rules. Because of errors in understanding of the examples it may take multiple opportunities before the correct production rule is formed. Also, further practice of a rule increases its strength and so the reliability of its application. We have done a number of experiments looking at different modes of tutoring which vary how long it takes subjects to go through a set of problems and the way in which they go through the problems. We find that what predicts students final achievement is how much practice they have had of these rules and not how that practice occurs At the extremes, we have compared students who are forced to always stay on a path of correct solution with students who are allowed to explore any path of solution they like with the only constraint that they finally execute the correct solution. Students in the second condition will take two to three times as long to get through the problems but show the exact same level of achievement. The basic implication of this principle is that one wants to minimize floundering time in problem solving and select problems which will offer practice on those productions where students most need practice. Much of the overall organizations of our tutors are motivated to maximize learning rate.

The fourth principle of tutoring that derives from ACT* concerns how to treat errors in student problem solving. Traditionally, much of the intelligence in intelligent tutors has been aimed at error remediation but we find little theoretical justification for extensive efforts at error remediation. There are four considerations which must be addressed in how one responds to errors: The first consideration is that many errors do not reflect misunderstandings or lack of knowledge on the students part but simply slips. The second consideration is the fact that people learn best when they generate the answer for themselves rather than are told (Anderson, 1990). The third consideration is that a great deal of time can be wasted in error states. The fourth consideration is that when students have problems with their knowledge what they need is another opportunity to learn the correct production (rather than a deep appreciation of their error). Thus, our recommended line of action is one in which we start out by simply pointing out an error to a student when it occurs without further comment. This avoids burdening the student with instruction when it is not needed, allows the student to self correct, and gives the students the opportunity to avoid going into an extended error state. We find that the majority of the time students are able to correct their errors without further instruction. When students cannot and request help, they are given the same kind of explanation that would accompany a training example. Specifically, we focus on trying to tell them what to do in this situation rather than focus on telling them what was wrong with their original conception. Thus, in contrast to the traditional approach to tutoring we focus on reinstruction rather than bug diagnosis. There has been some recent evidence that this is more effective (Sleeman, Kelly, Martinak, Ward, & Moore, 1989). This is not to say that our tutors are incapable of explaining what is wrong with the solution when asked. However, what they give at such points is more like an intelligible error message than it is like an attempt to debug a misconception.

Tutor Development

Within our system tutor development takes place in a characteristic sequence. In this section we would like to explain this sequence, what each step involves, the connection of each step to the general principles of tutor development, and what facilities we are providing to support that step of development. Right now this all takes place within a Macintosh Allegro LISP system and this will undoubtedly color our exposition. However, we hope to see our development system generalized beyond that platform.

Constructing a Tutor Interface

The first thing one must do in creating one of our tutors is to define the interface in which the problem solving is going to occur. This contrasts with the early tutors where we first developed the production system model and then searched for some tutor interface that would "realize" it. This shift in development approach reflects the observation that the interface defines in part the task that the student has to master and there is no definition of the skill independent of the interface. An interesting example concerns our tutors for teaching computer programming which use a structured editor interface that relieves the student of the burden of mastering the syntax of the programming language. The skill to be mastered in this interface is different than one which would require mastery of syntax and students who emerge from this tutoring do not know the syntax although they do not appear to have great difficulty in picking it up.

A number of factors impinge on the design of the interface. One involves decisions about what is important to master. In the case of computer programming we made a decision that syntax was not important and we wanted to focus on the semantics of the language. In general, if one is using a computer system for instruction it makes sense to use a relatively high-tech interface that does the mundane for the student and allows the student to focus on the more significant aspects of the domain. As a consequence our interfaces can serve as useful tools for problem solving even in the absence of a tutor. Another factor in designing an interface is to facilitate the process of model tracing in the tutor. In particular, it is useful to create an interface that involves a high density of rich input from the student to help us diagnose the student's problem-solving state.

The interface itself must be created in LISP although we have provided a number of facilities to assist the user. Creating an interface involves defining a number of interface interactions which have two side effects. One is to produce some transformation to a workspace on the screen. Thus, if a student selects "procedure" from a menu in our Pascal tutor and template for a Pascal procedure will appear. The second side effect is to deposit in the working memory of the production system a representation of this transformation of the screen. The consequence of this coupling is that the working of memory of the tutor always represents faithfully the problem state before the student on the screen. The actions in our tutor consist of either selecting menu items or typing values into slots created in the workspace.

Writing Production Rules and Model Tracing

The production rules for the system are defined in terms of the interface and its representation in working memory. The production rules match to states of the problem representation in working memory and call for interface actions. The actual problem representations contain more than just what is in the interface but also interpretations of these elements. Thus, production rule for programming might be, informally stated:

```
    IF the goal is to expand the output of a writeln statement
           and the output is the product of two numbers
THEN select "*"
           and pass the numbers to the arguments of "*".
```

The first clause "expand the output of a writeln statement" refers to a screen state where the writeln statement has not been completed. This is something of the form:

```
writeln(<output>);
```

"the output is the product of two numbers" represents an interpretation associated with the unexpanded argument, **<output>**. "select "*" refers to an interface action (select this from a menu). This will transform the code so that it looks like:

```
writeln(<arg1> * <arg2>);
```

"pass the numbers to the arguments of "*" " will serve to annotate **<arg1>** and **<arg2>** with the numbers they should represent.

When the tutor is in a model-tracing mode, it tries to find some sequence of production rules in its internal model that will match up to the behavior of the student. The tutor's goal in model tracing is always to account for the next action of the student. There may be multiple production rules that fire in the student model in order to make the transition between the current state and the next action. There are three possible outcomes when the tutor tries to interpret the next action of the student:

(1) A single sequence of productions is found which will produce that action. This then is taken as the interpretation of that segment of the behavior.

(2) Multiple sequences of productions are found. It is then not possible to interpret (model trace) subsequent behavior of the student until the ambiguity is resolved. This ambiguity can be resolved by presenting students with a menu of possible interpretations of their behavior and letting them determine the correct one.

(3) There is no interpretation of the student's behavior. The presumption is that the student's behavior is in error at this case. One thing we can do is to see if the behavior can be recognized by a buggy production sequence. Again until this behavior is changed, it is not possible to interpret (model trace) subsequent behavior.

What is actually done in states (2) and (3) depends on the choice of tutoring discipline. In immediate feedback mode the tutor will request an interpretation from the student in case (2) and force the student onto a correct path in case (3). However, in other modes it is possible to just suspend further interpretation and let the student continue to try to solve the problem in the interface. We only try to resolve the ambiguity in (2) or force a correct solution in (3) when the student requests further help from the tutor. We have looked at two versions of a less directive tutor. In one version, called flag tutoring, we resolve the ambiguities in (2) as they arise and warn the students of possible errors in (3) as they arise but do not force students to correct errors before going on. In another version, called demand-feedback tutoring, we neither try to resolve ambiguity or correct errors until the student requests help from the tutor.

One of the major technical accomplishments in our new tutoring architecture is that the production system is much more efficient than previous ones which allows us to search through multiple sequences of production rules looking for a match without perceptible pauses. The current system is also largely unaffected by problem complexity. In the case of programming, for instance, we have been able to tutor many-hundred line programs. We have also included a good many facilities to help the production system writer develop and debug the production rules.

Declarative Instruction

Once the production rules have been written we can then engage in the process of attaching to these production rules the declarative instruction which will help serve to communicate to the student the content of these rules. This instruction is meant to accompany the original example-based instruction or to provide correction in the context of a problem. Thus, the instruction is always example-based, with the example either being the training example or the test problem. The instruction takes the form of templates with slots which can be filled by English descriptions of the specific example. Thus, attached to the multiple production above might be a template of the form:

> At this point you want to write out the product of **<slot1>** and **<slot2>**. Use the symbol '*' to calculate a product.

where **<slot1>** and **<slot2>** corresponded to variables bound in the production. If in this example, the value bound to **<slot1>** had the English description "the employee's salary" and the value bound to **<slot2>** had the description "the applicable tax rate" this would become:

> At this point you want to write out the product of the employee's salary and the applicable tax rate. Use the symbol '*' to calculate the product.

Similar templates attached to error productions can be used to give context-specific error messages and descriptions in disambiguation menus.

As noted, there are two places at which we want to deliver such declarative instruction. One is when we are introducing the productions and the other is when the student needs reinstruction at a point of error. It turns out that these situations are not very distinct in practice. We actually deliver the original instruction in the context of a problem. One difference is that when a student gets to a point which requires instruction on a new rule, we simply volunteer the information right away; whereas, we wait until an error or request for help when it is an old rule. Another difference is that when we are helping the student at a point which they should know, we may want to initially provide less information and only later provide more direct information. Thus, we might first tell the student "At this point you want to calculate the product of the employee's salary and tax rate." Hopefully, this will be enough and the student will enjoy the benefit of partially solving this step. Only if necessary would we provide the additional "Use the symbol "*" to calculate the product." Thus, we have the facility to store successively stronger hints with a production. The first time a student is introduced to a production all of these hints are amalgamated into a unit of instruction.

In preparing declarative instruction, the developer must both attach these templates to production rules and associate English descriptions to elements of a problem. The template construction is a one-time investment but each problem entered into the system needs its own English embellishments. We have been working on facilities to expedite problem entry.

Knowledge Tracing and Mastery Instruction

There is a distinction between what we call model tracing and knowledge tracing. Model tracing refers to trying to interpret a student's solving of a single problem as a sequence of production rules in the tutor. From this attribution we can make inferences about how well the student knows the various productions. Knowledge tracing refers to trying to infer the growth of knowledge about the production rules across a sequence of problems. This is critical to optimizing the curriculum sequence. We can associate with each problem the production rules it involves. Our decision about what problem to next present to the student and when to promote the student onto new material is made in response to our interpretation of how well they know the various productions.

A Bayesian inference model for making such interpretations has been described in Corbett and Anderson (in press). Basically, it allows us to assign to each production rule a probability that the student knows that specific production. We select the problem that involves the maximal learning opportunity where this is defined as the problem with the highest density of productions with low probabilities. We continue presenting problems to students until we reach the point where the instructional gain (expected increase in their probabilities) for all the problems falls below a threshold defined by the length of the problem. Then we promote the student to the next unit in the curriculum sequence.

The overall curriculum is broken up into units where each unit involves the introduction of about a half dozen new productions. The first problem in each unit is meant to explicitly illustrate the new material and later problems are for purposes of practice. One of the tasks of the curriculum designer is to specify this division of the overall curriculum into units and specify the training problems. The other responsibility is to associate a number of parameters with the Bayesian inference scheme which define the difficulty of the various productions and the ability of the students. With this in place the system does a nice job of bringing students to mastery of the productions in each unit.

Summary

While the system is still in development, we have had considerable experience doing curriculum development with it in our lab and hope to make it available to other researchers. The importance of the production rule development comes strongly through in the experience of curriculum development. It is the component where most of the time is spent and it serves to organize subsequent development such as the design of declarative instruction. Interface design is separate but is largely a one-time initial investment whereas each unit of curriculum requires a fresh, full effort at production-rule modeling. Our system has made it easier to develop these rules and once developed provides a overall system in which they can organize efficient instruction. However, we see no way of relieving the developer of the essential question in our approach which is "How should the student think about solving this class of problems?"

This research was supported by Contract MDA-903-89K-0190 from the Army Research Institute and Grant 87-51890 from the National Science Foundation.

References

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, MA: Harvard University Press.

Anderson, J. R. (1990). *Cognitive psychology and its implications, Third Edition*. New York: W.H. Freeman.

Anderson, J.R., & Reiser, B.J. (1985). The LISP tutor. *Byte, 10,*159-175.

Anderson, J. R., Boyle, C. F., & Yost, G. (1985). The Geometry Tutor. In *Proceedings of IJCAI-85.* Los Angeles, CA: IJCAI, 1-7.

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive Modelling and Intelligent Tutoring. *Artificial Intelligence, 42,* 7-49.

Anderson, J. R., Boyle, C. F., Farrell, R., & Reiser, B. J. (1987). Cognitive principles in the design of computer tutors. In P. Morris (Ed.), *Modeling Cognition,* Wiley.

Anderson, J. R., Corbett, A., Fincham, J., Hoffman, D., & Pelletier, R. (In press). General Principles for an Intelligent Tutoring Architecture. In V. Shute and W. Regan (Eds.), Cognitive Approaches Automated Instruction.

Anderson, J. R., & Thompson, R. (1989). Use of analogy in a production system architecture. In S. Vosniadou and A. Ortony (Eds.), *Similarity and analogical reasoning,* 267-297. Cambridge University Press.

Corbett, A. T. & Anderson, J. R. (in press). The LISP intelligent tutoring system: Research in skill acquisition. In J. Larkin, R. Chabay, C. Scheftic (Eds.), *Computer Assisted Instruction and Intelligent Tutoring Systems: Establishing Communication and Collaboration.* Hillsdale, NJ: Erlbaum.

Koedinger, K. R., & Anderson, J. R. (1990). The role of abstract planning in geometry expertise. *Cognitive Science, 14,* 511-550.

Lewis, M. W., Milson, R., and Anderson, J. R. (1987). The teacher's apprentice: Designing an intelligent authoring system for high school mathematics. In G. P. Kearsley (Ed.), *Artificial Intelligence and Instruction*. Reading, MA: Addison-Wesley.

Singley, M. K., Anderson, J. R., Gevins, J. S., & Hoffman, D. (1989). The algebra word problem tutor, *Artificial Intelligence and Education*, 267-275.

Sleeman, D., Kelly, A. E., Marlinak, R., Ward, R. D., & Moore, J. L. (1989). Studies of diagnosis and remediation with high-school algebra students. *Cognitive Science, 13,* 551-568.

The Influence of Dialogue Processes on Students' Collaborative Generation of Explanations for Simple Physical Phenomena

Michael J. Baker
Centre National de la Recherche Scientifique, Laboratoire IRPEACS,
93 chemin des Mouilles, BP167 69131 Lyon-Ecully Cedex FRANCE.

Abstract: We present a method for analysing students' collaborative explanations for phenomena observed in simple mechanics experiments. Analyses have been performed from the points of view of the epistemology of the domain and the cognitive processes operating to generate the content of explanations. In this paper we concentrate on the influence of dialogue processes on processes described from these other two points of view. The dialogue processes considered are those of maintaining a shared focus, and generating utterances in the pragmatic context of high-level mutual interaction goals and complex propositional attitudes. We define two complex attitudes–COOPERATION and CONFLICT–and use these definitions to identify a number *cooperation* and *belief-conflict removal* strategies. Some reasons why explanations do not develop effectively are identified, and a number of ways in which computer-based guidance could be provided in collaborative distance learning are suggested.

Introduction

Current interest in the role of language (Vygotsky 1986) and inter-personal conflict (Doise, Mugny and Perret-Clermont 1975) in learning has stimulated the development of a new generation of computer-based learning environments which aim to facilitate collaborative work (Greif 1988; Beherend, Singer and Roschelle 1988; Chan and Baskin 1988). Our aim is to develop a method for analysing explanation dialogues between pairs of students to provide an empirical basis for designing effective intelligent guidance in collaborative computer-based learning at a distance. We describe how intelligent guidance in such a context could promoting constructive cooperation and conflict resolution which facilitates the emergence of effective explanations.

Although there is a considerable body of research on the epistemological content of students' conceptions in physics (Driver, Guesne & Tiberghien 1985), the cognitive processes of explanation generation (Schank 1986; Chi, Bassok, Lewis, Reimann and Glaser 1989) and the structure of explanation discourse (Goguen, Weiner & Linde 1983; Cawsey 1989), there has been little research on the influence of dialogue processes on explanation generation, notwithstanding Ericsson and Simon's (1984) model for language generation in protocol analysis. Closely related work is that of Beherend, Singer, and Roschelle (1988) and of Barnard and Erkens (1990) on the rôle of language in *problem solving* rather than *explanation*. Beherend, Singer, and Roschelle (1988) developed a methodology for analysing the "transaction level" in a collaborative physics learning environment, which identified when students were genuinely collaborating. In our method we analyse both COOPERATION and CONFLICT situations in terms of *complex propositional attitudes* (Fodor 1981), which enables us to describe how utterances may be generated from high-level cooperative goals, and to identify a number of dialogue strategies used in dealing with conflict.

Pairs of students were asked to cooperate in interpreting the phenomena observed in simple mechanics experiments, in terms of *energy*. A jointly agreed explanation should *emerge* (Miyake 1986; O'Malley 1987) from the interaction, which must be internally and externally *coherent* (Harman 1986) and *relevant* to the phenomena to be explained in terms of *intertranslatability* with descriptions at an 'objects and events' level (Chi, Feltovich and Glaser 1981). Students have *symmetrical* rights and high-level goals in the interaction, but there may be *differences* in lower-level goals and beliefs, which may lead to *conflict*.

Data Collection and Analysis

Dialogues (around 1½ hours long) were recorded and transcribed from students (n = 18) aged 17-18 years, working in pairs, from Lycees in Lyon. Observers were present, who made notes which were later used for clarification in analysis. Here we consider one specific dialogue ("Helene and Laurent"). Students were asked to identify which properties of balls enabled them to be distinguished in terms of their rebound behaviour, and to explain the phenomena in terms of energy. We expected the students' explanations to involve the concepts of *energy transfer* and *conversion*, *work*, *force* and various forms of energy (potential, kinetic, mechanical). Dialogues were coded in terms of the *epistemology* of the domain (Langlois and Tiberghien 1990), using referents at different conceptual levels (Chi, Feltovich and Glaser 1981), and at the *cognitive* level in terms of cooperatively generated production rules. We can not describe these analyses here, but our results are generally in accordance with those of Beherend, Singer and Roschelle (1988). This provides a basis for analysing the influence of dialogue processes on these levels, as described in the next section.

Analysis of Cooperation

We assume Perrault's (1990) formalisation of a number of Grice's (1975) maxims for cooperative conversations, a general "deliberative agent" architecture (Genesereth & Nilsson 1987), propositional attitudes derived from *Agent Theory* (*beliefs*, *intentions*, *goals*, *committments*, *preferences* and *wants*) and utterance categories derived from computational speech act theory (Cohen, Morgan and Pollack 1990). The analysis method consists of assigning the utterance categories and their propositional content to utterance segments, and showing how they may be generated from propositional attitudes and the high-level goals of the interaction. The complex attitudes defined for *conflict* should follow from definitions of the utterance categories. There are many profound unresolved theoretical issues in agent theory, concerning the nature of belief and intention, for example. Our research strategy is to take existing formalisations and to successively refine them with respect to analyses of real dialogue protocols. Here we can only give a short illustration of the model currently being developed.

The starting point for the analysis is the set of mutual goals constituting the "social contract" between the students:

L and H have the mutual goal (G1) that they will mutually believe that some set of propositions {p1, ... pn} explain their mutual beliefs in a given set of results (G2) {r1, ... rn} for a given set of experiments {expt_1,....expt_n}, that the propositions be instances of some subconcept of energy (G3). [L = Laurent; H = Hèlène]

G1 : (mgoal L H (mbelief L H (explains {p1, ... pn}
(rebound_behaviour (result {r1, ... rn} {expt_1,....expt_n}))))).

G2 : (mgoal L H (mbelief L H ({r1, ... rn} {expt_1,....expt_n}))).

G3 : (mgoal L H (mbelief L H (instance {p1, ... pn} (subconcept energy)))).

The definition of mutual goals ("mgoal") is derived from the work of Galliers (1989) and Elsom-Cook (1990):

R1 : (mgoal S1 S2 g) --> (goal S1 g) & (believes S1 (goal S2 g)) &
(goal S2 g) & (believes S2 (goal S1 g)).

We assume that the well-known infinite regress of mutual beliefs is simply stopped at the first level:

R2 :(mbelief S1 S2 b) --> (believes S1 b) & (believes S1 (believes S2 b)) &
(believes S2 b) & (believes S2 (believes S1 b)).

Let us now look at an example utterance pair:

Utterances	Analysis
"L: it rebounds all the same a bit lower ... [proposition]	(INFORM L H p1) p1 : (result (lower_than (rebound ball_1) (rebound ball_2)))
H: yes, but with respect to the masses you can see that the steel one is heavier"[1]	(EXPRESS H L (believes H p1))

Here we illustrate the approach by analysing a single utterance. To INFORM some proposition is to ASSERT it with the intention that the proposition be believed by the hearer (Perrault 1990). 'EXPRESS' is an utterance category where an *attitude* to some proposition is communicated, in this case, *belief* (Vanderveken 1990). Assume that a focus mechanism selects p1 as the most relevant proposition, both in terms of other propositions in semantic memory and in terms of the high level goals (Grosz and Sidner 1986). From G2, R1 and R2, and

R3 : (goal S1 x) --> (goal S1 (subcomponents x))

we can derive

(goal L (believes H p1)) &
(goal L (believes H (believes L p1)))

Other goals are derived which are already satisfied (such as that L should believe p1). If there are no other goals to which L is *committed* (Cohen & Levesque 1990) which he believes conflict with this goal, L may become committed to it. In this case, the agent *wants* to perform a dialogue action which it is believed will contribute to achievement of the goal state. Given the following definition, L wants to INFORM H of p1 [A1,A2 = agents ; x = proposition ; wants, believes = propositional attitudes ; INFORM = utterance category] :

R4 :(wants A1 (believes A2 x)) &
(wants A1 (believes A2 (believes A1 x))) &
(believes A1 x)
--> (wants A1 (INFORM A1 A2 x)).

Choice amongst wanted actions is made according to a *preference mechanism* which evaluates the extent to which an action is likely to be successful in achieving the wanted goal. Rules for inferring attitudes from utterances and for reasoning about the attitudes of others are therefore also required. We have not described many aspects of the model but have rather given a short illustrative example. The most important question to be addressed is how and when agents are willing to adopt beliefs stated by others (Harman 1986).

Analysis of Conflict

Since inter-individual conflict in problem-solving tasks has been argued to be a necessary condition for cognitive development, under certain specific conditions (Doise, Mugny and Perret-Clermont 1975), it may also play such a rôle in collaborative explanations. Results are, however, inconclusive, and conflict may in fact be associated with *less* progress (Blaye 1990). Our analyses suggest that the precise *dialogue strategies* used may be important in determining change in students' beliefs. Within "Distributed Artificial Intelligence" research, *negotiation protocols* have been described for resolving conflicts in goals (Sycara 1989) and task allocations between distributed agents. Little of this research has addressed the problem of resolving conflicts in *beliefs,* which has rather been approached from the perspective of *argumentation* (Ducrot 1980; Barth and Krabbe 1982; Alvarado, Dyer and Flowers 1986; Cohen 1987; Moeschler 1989). We define several forms of conflict in terms of complex propositional attitudes and the speech acts which they generate, based on modifications of the research of Galliers (1989) and Elsom-Cook (1990). This allows detailed analysis of the various ways in which conflict may be 'removed' or 'resolved'. Conflict *resolution* is associated with *belief revision*, and conflict *removal* with changes in *awareness* and *believed* belief revision. We also use descriptions of dialectic moves in argument - support, counterclaim, etc.–derived from the work of Toulmin, Rieke and Janik (1979) and Barth and Krabbe (1982). These do not constitute new utterance categories, but are rather *relations* between acts and between propositions and acts (Ferrara 1985).

For one agent to convince another to change their beliefs, they must be *aware* of 'external' (inter-agent) conflict, which must lead to awareness of 'internal' (intra-agent) conflict, which in turn, given a *coherence model* of *inconsistency* (Harman 1986), may lead to belief revision. We model "awareness" as focus of attention (Grosz & Sidner 1986) in working memory, in terms of "spreading activation" (Anderson 1983; Baker 1989b; Baker 1991a). *External conflict* concerns conflict between beliefs of two or more different agents. There are at least two different kinds–where some believed conflicting *alternative* proposition has been stated (ext_conflict_a"), and where there is simple *denial* ("ext_conflict_d")[2]:

(believes A1 X1) & (aware A1 X1) &
(believes A1 (believes A2 X2) & (aware A1 (believes A2 X2)) &
(believes A1 (inconsistent X1 X2))
--> (**ext_conflict_a** A1 A2 X1 X2).

(believes A1 X1) & (aware A1 X1) &
(believes A1 (believes A2 (not X1)) & (aware A1 (believes A2 (not X1)))
--> (**ext_conflict_d** A1 A2 X1).

External conflict may also be *mutual*, for example (similarly for **m_ext_conflict_a**):

(**ext_conflict_d** A1 A2 X1) & (**ext_conflict_d** A2 A1 X1)
--> (**m_ext_conflict_d** A1 A2 X1).

The process of resolving belief conflicts is one example of *negotiation* (Baker 1989a, 1989b, 1991a), which is not exclusively concerned with conflict resolution (Sycara 1989)[3], nor with conflicts in *goals* (Baker 1991a), but may rather also include the attempt to secure mutual belief from an absence of such.

Consider the definition of ext_conflict_d. This complex attitude is no longer ascribed to an agent if any combination of the conjucts in the definition is no longer ascribable ($2^4 - 1 = 15$ possibilities). For each specific conflict we therefore need to (i) ascribe the appropriate conflict attitude, (ii) describe which combination of conjunctive primitive attitudes no longer hold, and (iii) describe the dialogue processes which led to this. We give a few examples taken from a consecutive series of 39 exchanges near the beginning of the dialogue. The definitions are presently simple, but prove adequate for this dialogue.

Conflict example 1

[p_x = proposition ; plain text = students' utterances; **bold text** = analysis]

1 L: it rebounds all the same a bit lower, it's normal, but still ...
2 H: yes, but look, with respect to the masses, look you can see that the steel one is ... heavier
3 L: yes, but it's not a question of mass.
p_1 : (explains "mass"
(lower (rebound ball_1) (rebound ball_2)))
(ext_conflict_d L H p1)
4 H: well ; and , there's all the same potential energy involved, I'm sorry !
(ext_conflict_d H L p1)
(m_ext_conflict_d L H p1)
5 L: ...... ok, but if you've ...
6 H: if we ...
7 L: if you had a big steel ball, it would rebound ...
8 H: if we release them at the same height, so that there's one which has a greater mass than the other, the one with the bigger mass, would have greater potential energy ...
9 L: yes, but ...

10 H: so, there will be more ...
11 L: do you believe that if, if you have an enormous rubber ball like that, which is a kilogramme, you believe it would rebound a lot?
12 H: yes, but that's only valid in the case of a perfectly elastic impact.
13 L: right, ...
14 H: in fact, I think ...
15 L: perhaps we'd be better off thinking about that, because, a priori, it's simpler, given that it's an inelastic impact

At 3 and 4, L denies H's (indirect) claim that the difference in rebound behaviour is explained by the *mass* of the balls. Since L does not propose an alternative explanation, this is analysed as **ext_conflict_d**, which becomes **m_ext_conflict_d**. From our definition of **ext_conflict_d**, minimal attitudes of H and L are :

Helene: (believes H p1) & (aware H p1) &
(believes H (believes L (not p1)) & (aware H (believes L (not p1)))

Laurent: (believes L (not p1)) & (aware L (not p1)) &
(believes L (believes H p1)) & (aware L (believes H p1))

Mutual conflict is transformed to individual conflict when any combination of the conjuncts for either H or L ceases to hold, and mutual conflict is removed when any combination of conjuncts no longer holds in the complex attitude for each. The strategies used here are:

- since L has produced no direct successful counterargument (11) against H's supports (4, 8) for p1, H *assumes* that L accepts p1
[(believes H (believes L p1))]
(as shown later in the dialogue, in fact (believes L (not p1))) ;
- the *focus of attention* shifts to considering elastic and inelastic impacts (12-15), giving
[(not (aware L (not p1))) & (not (aware H p1))] (conflict removed);
- the *cognitive resources* which H and L are willing to use in resolving this conflict are *expended*, and so a focus shift is explicitly signalled (15).

Conflict example 2

28 H: yes, elastic impact, there's the total energy which is conserved.
29 L: yes ...
30 H: yes, but there's the friction ... of the air!
    p2: (explains (friction air)
        (result (not (same (rebound ball_1) (rebound ball_2)))))
31 L: oh, I don't think its especially the air friction which invervenes ...
    (ext_conflict_d H L p2)
32 H: but yes, otherwise it would rebound to the same height!
    (m_ext_conflict_d L H p2)
    p3 : (same (rebound ball_1) (rebound ball_2)) [stated earlier in dialogue]

p4: ((((not p2) --> p3)) & (not p3)) -> p2)
33 L: no! ...
(ext_conflict_d L H p4)
34 H: but yes!
(m_ext_conflict_d L H p4)
35 L: ... it's the loss at the moment of impact ...
p5: (explains loss_at_impact
(result (not (same (rebound ball_1) (rebound ball_2)))))
36 H: that's the same ... that's really a sort of friction also. It's a form of friction, either with the ground or with the air
(ext_conflict_a L H p5)
p6: (equivalent loss_at_impact (friction air))
37 L: yes, but after all, air friction in comparision with friction ... no, no, what you call ground friction is by contrast negligable!
(ext_conflict_a L H p6)
38 H: ah well .... yes

Here we have a succession of conflicts. We can not even assume that the final one is resolved (H's "yes," line 38) since H may be simply "taking her turn" in the dialogue. In the phase 28-34, conflicting propositions are *denied*, and in 35-38 alternative propositions ("counterclaims") are proposed. Both awareness and belief are involved in removal of the conflicts. Firstly, L assumes that H no longer believes the conflicting propositions, since he believes this last argument to be *won*, and secondly, neither student is now *aware* of the original propositions with respect to which conflict took place. The dialogue strategies used here in *belief-conflict removal* are summarised as follows:

focus shifts : focussing on
- related propositions with respect to which there is agreement;
- most recent conflicts;
- most recent 'won' local arguments;
- (focus shifts are sometimes occasioned by *exhaustion of cognitive resources* when conflict is not resolved after a certain time);

belief revision assumptions: assuming that
- 'won' local arguments imply belief revision;
- unsupported alternative claims imply acceptance of supported claims;
- absence of alternative claims implies acceptance;

lack of clarity
- students' understanding of their own and others' arguments and beliefs is unclear, which leads to an unclear awareness and understanding of the conflict situation.

It is understandable that conflict between students tends to be *removed* rather than *resolved* and that beliefs are not genuinely revised in such conflicts: in the absence of an arbiter, such as a human teacher, there is no direct *legitimisation* of their beliefs, although students do employ various strategies for *validating* and/or *verifying* their results and explanations (Langlois 1990), with respect to their own knowledge, and aspects of the general "didactic contract" ("all information in the question is relevant," "the teacher said so," etc.). We hypothesise the following rôles for automatic intelligent guidance in facilitating students' explanations in computer-based collaborative distance learning in conflict situations:

- clarification of conflicting propositions;
- clarification of beliefs of each student;
- identification of dialogic or argumentative relations between propositions;
- recapitulation and reinforcement of argument conclusions;
- indication as to when an argument is worth pursuing.

The generation of effective collaborative explanations between students therefore requires guidance on *how to argue*.

Conclusions and Further Work

We have described a computationally tractable approach to analysing students' collaborative explanations, which aims to identify the influence of dialogue processes. This may help us to identify reasons why explanations do not develop effectively, and thus to define a rôle for computer-based guidance in the context of collaborative computer work. Our analysis suggests that *cooperation is more constructive than conflict in the evolution of students' explanations*. There is a clear need for guidance in *cooperative* contexts, however, since students may agree on explanations which, in the absence of legitimisation, are not the most effective which have been stated. We also identified ways in which students 'remove' rather than resolve conflicts, and some possible rôles for guidance. In future work we shall apply the exploratory analysis method developed to other dialogues. We speculate that using a deliberately *restricted* communication interface in computer-based collaborative learning —such as that described by VanLehn (1985) for arguments—may help to reduce some of the ambiguity of explanations and arguments.

Notes

1. All quoted dialogues have been translated from the original French by the author.
2. Here we differ from Galliers (1989) and Elsom-Cook (1990).
3. Sycara (1989) restricts negotiation to "...the process of achieving compromise between conflicting goals."

Acknowledgements: Many thanks to Andree Tiberghien for constructive criticism throughout this research, and to Françoise Langlois for generously making the dialogues available.

References

Alvarado, S.J., Dyer, M.G. and Flowers, M. (1986). Editorial Comprehension in OpEd through argument units. *Proceedings of the First National Conference on Artificial Intelligence*, 250-256, Philadelphia USA.

Anderson, J.R. (1983). *The Architecture of Cognition*. Harvard: Harvard University Press.

Baker, M.J. (1989a). A model for tutorial dialogues based on critical argument. *Artificial Intelligence and Education*, (eds.) Bierman, D., Breuker, J. & Sandberg, J. Amsterdam: IOS Publishers.

Baker, M.J. (1989b). *Negotiated Tutoring: A Model for Interaction in Intelligent Tutoring Systems*. PhD thesis in Cognitive Science, (unpublished). Institute of Educational Technology, Centre for Information Technology in Education: The Open University, Great Britain.

Baker, M.J. (1990a). Arguing with the tutor: a model for tutorial dialogue in uncertain knowledge domains. In *Guided-Discovery Tutoring*, (ed.) Elsom-Cook, M., p. 141-170. London: Paul Chapman Publishing.

Baker, M.J. (1991a, in press). Negotiating Goals in Intelligent Tutoring Dialogues. To be published in *New Directions in Intelligent Tutoring Systems*, (ed.) Costa, E., NATO ASI Series.

Baker, M.J. (1991b, in press). Negotiated Explanations in Students' Physics Dialogues. To be published in *Knowledge Acquisition in Physics and Learning Environments*, (eds.) Tiberghien, A. & Mandl, H., Springer-Verlag, NATO ASI series.

Barth, E.M. and Krabbe, E.C.W. (1982). *From Axiom to Dialogue: A Philosophical Study of Logics and Argumentation*. Berlin: Walter de Gruyter.

Barnard, Y. and Erkens, G. (1990). Simulation and Analysis of Problem-Solving and Dialogue Processing within Co-operative Learning. *Learning and Instruction : European Research in an International Context.* Volume 2:1, Social and Cognitive Aspects of Learning and Instruction, (eds.) Mandl, H., De Corte, E., Bennett, S. N. & Friedrich, H.F., pp. 181-196. London: Pergamon Press.

Beherend, S. Singer, J. and Roschelle, J. (1988). A Methodology for the Analysis of Collaborative Learning in a Physics Microworld. *Proceedings of ITS '88*, 48-53.

Blaye, A. (1990). Peer Interaction in Solving a Binary Matrix Problem : Possible Mechanisms Causing Individual Progress. In *Learning and Instruction* Vol 2,1, (eds.) Mandl, H., De Corte, E., Bennett, N. & Friedrich, H.F., pp. 45-56. Pergamon Press.

Cawsey, A. (1989). *Generating Explanatory Discourse: A Plan-Based, Interactive Approach*. PhD thesis (unpublished), University of Edinburgh.

Chan, T.W. and Baskin, A.B. (1988). "Studying with the Prince": The Computer Companion as a Learning Companion. *Proceedings of ITS-88*, Montrèal.

Chi, M.T.H., Bassok, M., Lewis, M.W., Reimann, P. and Glaser, R. (1989). Self-Explanations: How Students Study and Use Examples in Learning to Solve Problems. *Cognitive Science* 13 (2), 145-182.

Chi, M.T.H., Feltovich, P.J. and Glaser, R. (1981). Categorization and Representation of Physics Problems by Experts and Novices. *Cognitive Science* 2, 121-152.

Cohen, R. (1987). Analyzing the Structure of Argumentative Discourse. *Computational Linguistics,* 13 (1-2), 1-24.

Cohen, P.R. & Levesque, H.J. (1990). Intention is Choice with Committment. *Artificial Intelligence* 42, 2-3, 213-261.

Cohen, P.R., Morgan, J. and Pollack, M.E. (1990). *Intentions in Communication*. Cambridge, MA: MIT Press.

Doise, W., Mugny, G. and Perret-Clermont, A.N. (1975). Social Interaction and Cognitive Development. *European Journal of Social Psychology*, 5, 367-83.

Driver, R., Guesne, E. and Tiberghien, A. (1985). *Children's Ideas in Science.* Milton Keynes (GB): The Open University Press.

Ducrot, O. (1980). *Les Echelles Argumentatives*. Paris: Minuit.

Elsom-Cook, M. (1990). Analysis of a Tutorial Dialogue. In Elsom-Cook, M. (ed.) *Guided Discovery Tutoring*, p. 113-131. London: Paul Chapman Publishing.

Ericsson, K.A. and Simon, H.A. (1984). *Protocol Analysis: Verbal Reports as Data.* Cambridge, MA: MIT Press.

Ferrara, A. (1985). Pragmatics. In *Handbook of Discourse Analysis, Vol. 2: Dimensions of Discourse*, (ed.) Van Dijk, T. London: Academic Press.

Fodor, J.A. (1981). *Representations.* Cambridge, MA: MIT Press.

Galliers, J.R. (1989). *A Theoretical Framework for Computer Models of Cooperative Dialogue, Acknowledging Multi-Agent Conflict.* PhD thesis (unpublished), Human Cognition Research Laboratory, The Open University (GB).

Genesereth, M.R. & Nilsson, N.J. (1987). *Logical Foundations of Artificial Intelligence.* Los Altos, Calif. : Morgan Kaufmann.

Goguen, J.A., Weiner, J.L. & Linde, C. (1983). Reasoning and Natural Explanation. *International Journal of Man-Machine Studies* 19, 521-559.

Greif, E. (1988). *Computer-supported cooperative work: A book of readings.* Morgan Kaufmann.

Grice, P.H. (1975). Logic and Conversation. In *Syntax and Semantics, Volume 3: Speech Acts*, (eds.) Cole, P. & Morgan, J.L. New York: Academic Press.

Grosz, B.J. and Sidner, C. (1986). Attention, Intentions and the Structure of Discourse. *Computational Linguistics*, 12(3), 175-204.

Harman, G. (1986). *Change in View.* Cambridge, MA: MIT Press.

Langlois, F. (1990). *Problemes de Physique en 1oS: Etude de Quelques Criteres de Verification.* Memoire de DEA, dir. A. Tiberghien, Groupe Didactique de Physique, Universitè Lyon I.

Langlois, F. and Tiberghien, A. (1990). *Resolution de Problemes de Physique par des Eleves des Lycees.* Rapport Interne Laboratoire LIRDIS (Universite Lyon-I) et Laboratoire CNRS-IRPEACS.

Larkin, J., Reif, F., Carbonell, J. and Gugliotta, A. (1988). FERMI: A Flexible Expert Reasoner with Multi-Domain Inferencing. *Cognitive Science* 12 (1), 101-138.

Miyake, N. (1986). Constructive Interaction and the Iterative Process of Understanding. *Cognitive Science* 10 (2), 151-177.

Moeschler, J. (1989). *Modèlisation du Dialogue.* Paris: Hermes.

O'Malley, C. (1987). *Understanding Explanation.* Cognitive Science Research Report CS-88, University of Sussex (GB) Department of Cognitive Studies.

Perrault, R. (1990). An Application of Default Logic to Speech Act Theory. *Intentions in Communication*, eds. Cohen, P.R., Morgan, J. & Pollack, M.E. Cambridge, MA: MIT Press.

Schank, R.C. (1986). *Explanation Patterns.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schegloff, E.A. and Sacks, H. (1973). Opening up Closings. *Semiotica.* 8 (4), 289-327.

Searle, J. (1969). *Speech Acts: An Essay in the Philosopy of Language.* Cambridge: Cambridge University Press.

Sycara, K. (1989). Multiagent Compromise via Negotiation. In *Distributed Artificial Intelligence: Volume II*, (eds.) Gasser, L. & Huhns, M.N. Pitman Publishing, London, pp. 119-137.

Tiberghien, A. (1990). Learning and Teaching at Middle School Level of Concepts and Phenomena in Physics: The Case of Temperature. In *Learning and Instruction* Vol 2,1, (eds.) Mandl, H., De Corte, E., Bennett, N. & Friedrich, H.F., pp. 631-648. Pergamon Press.

Toulmin, S., Rieke, R. and Janik, A. (1979). *An Introduction to Reasoning.* New York: Macmillan.

Vanderveken, D. (1990). On the Unification of Speech Act Theory and Formal Semantics. In *Intentions in Communication*, eds. Cohen, P.R., Morgan, J. & Pollack, M.E., 195-220. Cambridge, MA: MIT Press.

VanLehn, K. (1985). *Theory Reform Caused by an Argumentation Tool*. Xerox Palo Alto Research Center Report P85-00102.

Vygotsky, L. (1986). *Thought and Language*. Revised & edited by Kozulin, A. London: MIT Press.

The Evaluation of Innovative Computer-Based Learning Systems: Teaching Newtonian Mechanics

S.G. Bento
Institute of Educational Technology
The Open University, Walton Hall
Milton Keynes, MK7 6AA, United Kingdom

Abstract: The research discussed in this paper outlines recent empirical research on the evaluation of a prototype Innovative Computer-Based Learning system. Before proceeding with an evaluation a framework should be provisionally defined. Initially the domain is important to examine from a theoretical perspective; the type of evaluation methodology, and prevailing legal requirements. The example problem is a second order predicate calculus question, where 9 to 10 year olds seek the solution by engaging the prototype innovative computer-based learning system. The discussion examines the communicable categories of children's apparent problem solving strategies as well as problems in questioning. Future improvements to the prototype are suggested

Introduction

The evaluation of Innovative Computer Based Learning systems examines the teaching or pedagogical process between the ICBLs and the student. ICBLs may include CAL, microworlds or systems that incorporate artificial intelligence (e.g., Scanlon and O'Shea 1986). Pea 1987 for example suggests that microworlds supports student's cognition to engage in problem solving. An integration between computer intelligence and human intelligence may therefore be achieved. Before evaluating an ICBLs several areas need to be considered. These include negotiations with the teacher and staff of the school; prevailing legal requirements (e.g., Education Reform Act 1988), the domain, and the type of evaluation methodology.

The Domain

In the UK physics is taught under the general heading of science. Until recently many elementary schools were under developed for helping children understand basic concepts in physics and science (DES 1989). Furthermore, little consideration has been given to using and evaluating ICBLs for educational use in the classroom.

The curriculum for elementary science in the UK usually makes a brief reference to the classic experiment about forces acting on falling bodies. This was supposed to have been performed by Galileo. Perhaps the experiment is mythical, however it marked a dramatic turning point in history, where the commonly accepted axiom of Aristotle that falling bodies were regulated by their respective weights was refuted (Cooper, 1935). The point illuminated by Galileo's refutation was that the blind prejudice of an unreasoning hero worship was too strong even for the evidence of the senses of sight and sound. Darrow (1930) remembers Aristotle chiefly as the perpetrator of one of the most colossal blunders in the whole history of science. This is important because the expectation that children engage in general Aristotelian misconceptions may be too simplistic. Obvious parallels may also be drawn. Children who are inadequately taught or provided with little opportunity to explore their notions of fundamental physics may live in a state of ignorance. The axiom "the ignorance of the laws of motion is the ignorance of the laws of nature" may be applied here in the truest sense. Until recently young children had little opportunity to test their ideas about physics. The microworld evaluated, may aptly provide and represent the in depth historical and scientific debate between Aristotelian notions of physics against Galilean and Newtonian notions. It is this kernel of science that "Friction Microworlds" may allow children to investigate and experience for themselves. However, the task of the evaluation is to establish how well the microworld provides pedagogical support for the domain of qualitative physics.

If a child is asked about her knowledge in physics she may deny having knowledge in the area. So there is a need for careful questioning and detailed teaching. In the evaluation of ICBLs used to teach physics the domain may be seen in two spheres. Qualitative physics and quantative physics. The former may be seen to involve a qualitative calculus; the recognition of qualitative propositions, visual proofs, combinatorial problems-eg predicate calculus. First order logic gives inner structure for atomic visual sentences which express the relationships between things. ICBLs should provide a means for children to express with generalisations and say that certain properties holds for all objects, or no objects. Quantative physics differs from qualitative physics in that it uses symbols such as $F/M = G$. An important question is what is the difference between such approaches. Qualitative physics allows observation, more flexible description, whereas quantative symbolics taught at an early age may not allow the same level of understanding as that allowed by qualitative physics. Qualitative physics allows the understanding of physical systems through a dialectical exploratory and didactic process, also through the representation of detailed theoretical models. The ICBLs evaluated in this instance should allow children to make generalisations, more complex empirical observations and seek answers to questions such as Galileo's "Tower of Pisa" problem (also see De Kleer et al., 1990; Forbus, 1990; Kuipers, 1990).

One important question in the domain is do the children exhibit general Aristotelian misconceptions? For example Disessa (1982) found that when students were asked to control Newtonian objects they exhibited general Aristotelian expectations that things should move in the direction that they were last pushed. This contrasts with the Newtonian expectation that force correlates with changes in velocity, or change in position. Viennot (1979) also suggests that 50% of students exhibit Aristotelian expectations that things go in the direction that they are last pushed.

Evaluation Methodology

The evaluation of an ICBLs needs to go through several processes. A holistic model has been proposed (Bento 1990). The first is a period of negotiation with the teachers of the school. As pedagogical environments are sometimes uncertain and complex the method of hermeneutics can be useful. Usually such method is applied to the interpretation of sacred texts. However, such methods can be extended to the interpretation of the academic text and sometimes the immediate context. In this case it was a middle school environment. Comparisons may be made between the histories and present horizons. For an introduction see Gadamer (1976); Winograd and Flores (1988); and Kemmis (1976). In a sense the hermeneutic method allows a deeper and illuminative interpretation of meaning than what is sometimes apparent.

In this study protocol analysis exemplified by Ericsson and Simon (1984) is used. Such a method allows the recording of verbal problem solving activity by students using an ICBLs, strategies and evidence of their final conclusion. However, verbal dialogue may not always reflect what is thought (Nisbett and Wilson, 1986). Where verbal protocol analysis is ineffective then video protocol analysis may be used. A students sensori-motor actions to an ICBLs may be recorded and analysed for apparent problem solving structure.

Ethics

One of the most important aspects to be realised with these methods is the observation of ethics. This includes respect for the individual and the freedom of the child; person not engaging in deception or the denial of the humanity of the person. The use should be free from political, ideological, and personal abuse. In brief such methods are used to help people freely learn concepts, not for a means of pure social or ideological control.

Example Problem

A second order predicate calculus question or second order logic question is presented. An example is the following statement-"There is a set of elements in this group whose characteristics are exclusive to other elements in the same group." As this sentence refers to a sub set with certain exclusive properties, it is thus a second order statement. Using the Friction Microworlds; middle school students between the ages of nine and ten were presented with the following second order problem. Their problem solving task was to determine which planets rules of motion resemble those rules of motion that would be found on Earth minus air resistance.

Friction Microworlds

Friction Microworlds is an Innovative Computer Based Learning system (ICBLs) originally developed and tentatively evaluated by Spensley et al. (1990). The ICBLs allows dyads to engage in an imaginary journey through seven nominal planets. Six of the nominal planets have conjectural rules of motion, with one planet being the solution planet. That is with the rules of motion that would be found on Earth minus air resistance. They start the journey by selecting a nominal planet to make empirical observations within, using the planet selection interface (see Figure 1). By pressing on one of the selectors, in response to the "Where to Boss" message-the dyads can commence on their journey through any planet and make experimental observations about each planets' rules of motion, and match them with their own reasoned beliefs about the solution to the problem.

On arrival the dyads can dispense a solidifying substance; sand, syrup or ice onto the planet surface (see Figure 2). This is achieved by using the mouse to direct the "hand" screen cursor over one of the dispenser buttons. Blocks are then individually grabbed from the ledge and dropped from a metaphorical height of 200 metres. To facilitate the dyads in their qualitative reasoning the indicators in the simulation provides numerical data on the

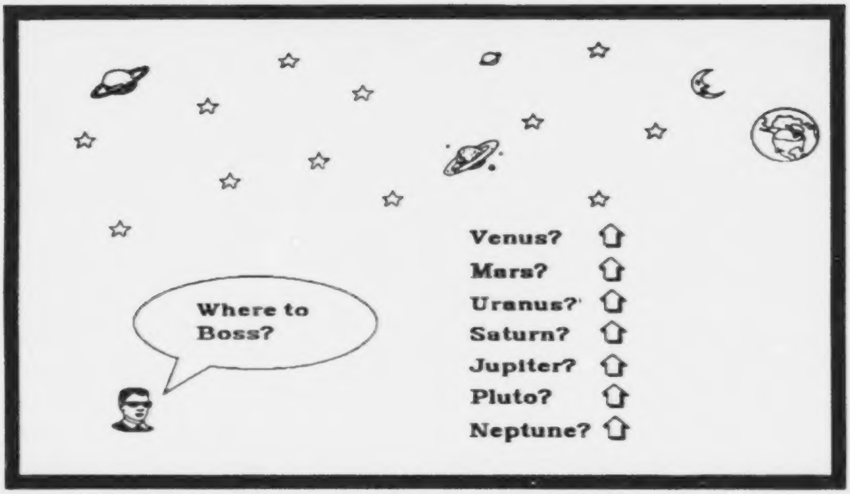


Figure 1. The planet selection interface for Friction Microworld

height of each block dropped, time to complete descent, velocity, and momentum. A horizontal meter provides the dyads with the distance travelled of each block. If the dyads wish to make a new set of observations using a different substance a reset button is provided-returning all the blocks to the ledge and all the indicators are re-initialised. The "telephone" object allows the dyads to "call up" for another planet and engage in further empirical observations. The "hoover" object allows the dyads to clear the planet surface and "lay" one new surface substance, perhaps experimenting later with different substances. The simulation objects and actions also provide the dyads with realistic visual and aural feedback.

Six of the planets have conjectural rules of motion-based on a blocks surface area and mass. The six could be termed as counter-examples or foils. The astute student should identify Pluto as the solution planet and give the correct solution, partial explanation, or full justification. The seven planet's rules are as follows:

Planet	Rules of Motion
Uranus:	Inversely proportional to mass and surface area
Jupiter:	Proportional to surface area
Venus:	Inversely proportional to surface area
Saturn:	Inversely proportional to mass
Mars:	Proportional to mass
Pluto:	Earth, minus air resistance
Neptune:	proportional to mass and inversely proportional to surface area

Solution: In the solution planet all the blocks travel the same distance because their is no air resistance (Newton law 1,2 ¬3), so there is no upward force when the blocks are dropped. The only force acting on the blocks is acceleration due to gravity which is constant 9m/s (Newton law 1). Therefore all the blocks have the same acceleration due to gravity and momentum subsequently landing on the surface with equal force. The coefficient of friction is the opposing force and equal for all blocks regardless of mass and surface area (Newton Law 3). Put very simply the child should recognise that the force applied on the blocks going down the frictionless chute is constant, and the opposing kinetic friction is constant (see Figure 2). This is the solution to the main problem. Subsidiary problems were also set involving the children describing the differences between substance's relative friction during evaluation sessions.

Case Study Methodology

Equipment

The equipment the evaluator used in this study consisted of an Apple Macintosh IIcx computer, one Sony Walkman professional tape recorder to record the concurrent protocols, solution, and subsequent justification of the dyads. The software was the 'Alternate Realities' microworld originally developed by Spensley et al. (1990) for assisting in the teaching of horizontal motion.

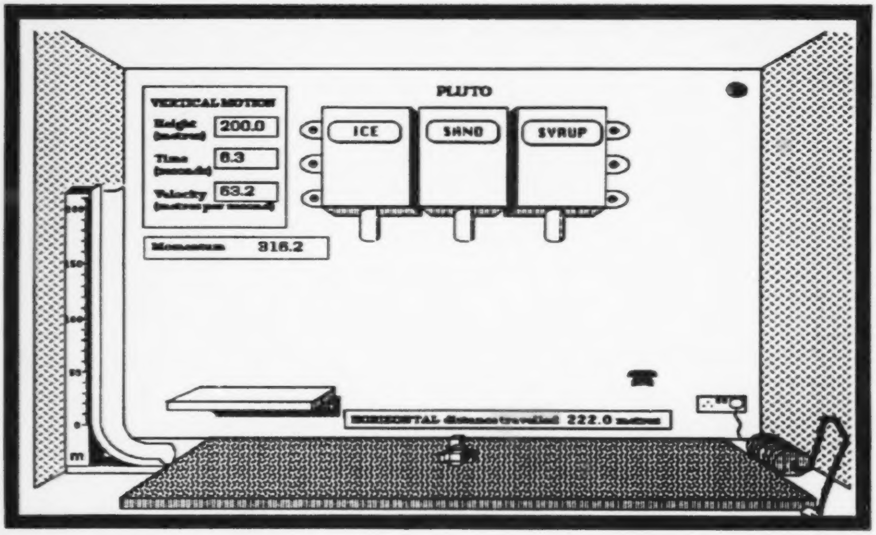
The subjects in the main evaluation study consisted of twenty-eight 9 to 10 year olds, 14 boys and 14 girls. The notation to depict the characteristics of the subject are used as follows: For example B1.9; b = boy, I = subject number, and the last number denotes the subjects age. Subject codes with G to denote girl. This notation is used to ensure relative confidentiality and subjects' anonymity.

Conditions

The evaluation took place in between two adjacent classrooms. Dyads came from one of the adjacent classrooms to use Friction Microworlds. The immediate environment was open plan, with an innovative middle school background environment. The only major interruption during the evaluation sessions was that the dyads took a 15 minute recess along with rest of the school.

Procedure

The evaluator set up a position in between two classrooms and received dyads from one classroom. Each dyad pedagogically engaged the microworld for an average of two hours. It took seven weeks of three half day periods to get through 14 dyads. There was an introductory discussion centred on the dyads personal characteristics, experiences in science, and previous computer use. At the beginning of the evaluation the teacher took an active part, but this participation receded because of her commitments to her class. Once the introductory discussion had finished the dyads were introduced to the operation of the computer hardware and interface. The dyads were then left to explore the ICBLs and experiment.



Height (metres) 200.0
Time (seconds) 6.3
Velocity (metres per second) 63.2
Momentum 316.2
HORIZONTAL distance travelled 222.0 metres

Figure 2. The solution planet where the rules of motion applied to the blocks resemble those rules of motion that would be found on Earth minus air resistance

The evaluator ensured that the dyads could understand the gross characteristics of the microworld and presented them with the target problem solving task. Each dyad before pedagogically engaging the microworld were asked to find out which planets rules of motion resemble those rules of motion that they would reasonably expect to be found on Earth minus air resistance. They were to give the evaluator the solution planet when they had made empirical observations on all of the planets. A major intervention before the dyads commenced with the problem, involved the evaluator guiding the dyads to drop the blocks at a height of 200 metres. Other interventions made by the evaluator included questioning about their observations about the rules of motion on each planet, and questioning them about the relative friction between ice, syrup, and sand. No direct cues were provided to the dyads to help them determine the solution planet.

Results

The qualitative results taken from the protocols are described on the next page in the form of a grid (see Figure 3). Each dyad number is given, subject code, answer (solution planet) and abridged justification. Sample abridged justifications are subsequently given for dyad 7, 10, and 11.

Dyad	Subject Code	Answer	Justification
1	B1.9	Uranus	Little block should come first.
	B2.10		
2	G1.10	Venus	Upright blocks have more friction
	G2.10		
3	G3.9	Pluto	All in heap
	G4.10		
4	B3.9	Venus	Venus closest to Earth
	B4.10		
5	G5.9	Pluto	Pile up behind each other
	G6.10		
6	B5.10	Pluto	Scarce
	B6.9		
7	G7.10	Mars	Perfect qualitative description
	G8.9	Pluto	
8	B7.10	Mars	Procedural misconception
	B8.10		
9	G9.10	Saturn	Small go first
	G10.10	Mars	
10	G11.10	Saturn	Strange if small went first
	G12.10	Uranus	
11	B9.10	Pluto	Paradoxical/Galileo
	B10.10		
12	B11.10	Mars	Small don't go very far
	B12.10		
13	G13.10	Pluto	Must be travelled together
	G14.10	Saturn	
14	B13.10	Pluto	Same gravity pull
	B14.10		

Figure 3. A grid of dyads expected solution planets and abridged justifications

Dyad 7

163 E: What causes...you said two things...can you describe where gravity and friction [as you said before] is happening?

164 C: Well, the gravity could pull it down but it could travel a little way....but the friction and the gravity would both sort of pull it together....and that stop it...but it would be the same with all of them...it would be the same amount.

Dyad 10

This dyad chose Saturn and Uranus...G12.10 chose Uranus

253 Em: That one I would say Uranus because 28 is the lowest...way low....I'd say Uranus...On one of them the longer block came 26 which was probably...say if the smaller one went 28 it would be strange because it was not that heavy.

G11.10 thought: Cause they're all sort of low numbers and there's only one over 100.

[clear procedural misconception]

Dyad 11

They chose Pluto as the solution planet with the justification being

236 E: Why do you think Pluto
237 I: Because they all fall the same and um..

[Galileo versus Aristotle Debate]

247 I: On the iron aqueduct...do you know the iron aqueduct?
248 E: Sorry
252 I: I went there yesterday I dropped a stone then dropped a leaf and they went the...same amount of time down.
254 I: I dropped it.....I dropped the stone and it went 4 seconds...and I dropped the leaf and it went 4 seconds.
262 I: What like time.....what I saw was the leaf kind of like went like that....and the stone kind of dropped straight down...they went the same time......

Discussion

The children who do not arrive at the correct answer and justification may be seen to have complex procedural misconceptions. That is they do not recognise the combined effect of the application and absence of the various forces. An imbalance attention to gross characteristic of the blocks is evident. Figure (3) indicates that the solution planet does not appear to be chosen at random or through novelty effects. When the abridged transcripts are examined in detail several types of problem solving or thinking may be apparent (see Bento, 1991 for abridged transcripts of all dyads).

1. Misconception

The child fails to recognise where all the various forces are applied and focusses primarily on the gross characteristics of the blocks. Declarative focus rather than procedural. This is exemplified by the dyad 10 transcript.

2. Sensory Concept Deficit

In this categorisation the 'remedial' child appears to concentrate on the gross characteristics of the microworld such as the hoover, slide, substances, enjoying mainly the visual and aural effects.

3. Misinterpretation of the Intended Parameters of the Question

Misinterpretation of the question and the microworld (Friction Microworlds) not perceived as one may wish. For example the child may start talking about the dimensions of space and the movement of the planets. However misconception cannot be concluded.

4. Epistemological Determination

This is where thought is epistemological in nature and the child contemplates the epistemological nature of questions. After going through the microworld. Dyad 11 thought about the nature of their applied knowledge. Just like the questions raised by Galileo the child had dropped a leaf and a stone off an iron aqueduct, wondering if one object would land before the other. Then the child separately drops each object and times its descent. Once performing this private experiment the child brings this previous knowledge to bear and determines the correct solution to the microworld problem. A clear example of ICBLs maximising and capturing a child's epistemological experience, as well as an explanation.

5. Higher Order Calculus

This is a clear example of the child apparently reasoning on a comparable level of an expert physicist and/or mathematician. In this dyad there appears to be a full understanding of the application and absence of forces at the age of nine. Through the decomposition of various forces; first order logic is displayed. Through recognising exclusive rules within a set of conjectural rules second order predicate calculus may be seen to have been achieved. However, this example as shown on the previous page indicates that requesting the answer is precarious because of the false negative phenomenon.

Conclusion

This study indicates that the contention that nine to ten year old children engage in general Aristotelian thinking about physics problems may be too simplistic. Indications are of complex procedural misconceptions that may be broken down into finer terms including misconception, sensory concept deficit, epistemological determination, and higher order calculus. There are obvious implications for serious science teaching of young children. Friction Microworlds has been evaluated as a viable prototype. Improvements may be made to the prototype to help young children understand qualitative calculus more explicitly. One of the contributions of this research is to show the importance of evaluation. Although the following is not presented as substantial evidence, it is provided to the reader to suggest and indicate that the teacher or part of the educational system may even have some (perpetual) misconceptions about the domain which it is teaching children. This is exemplified from the following case study, where a 10 year old (with his father present) incorrectly justifies his answer.

J: Saturn-the longer ones don't go as fast.
J: The heavier ones will go done faster.

E: So what you're saying is that the longer ones will go down the chute quite fast?

[he thinks about it]

J: Measure weight-if you drop a brick and an apple. The brick will go down first. Because the block was 10 [newtons]. Say the apple was 10 kilograms....I don't know if that right or not..it would have 100 Newtons.

E: Force?
J: Yeah

J: And the brick would have a bigger force and go faster...but it would land straight on the bottom really...the apple might go a little further.

J: The brick is so heavy it goes to land..the apple goes to land and bounces back up.

E: So which lands first

J: The brick
E: Why's that?
J: Because its heavier
E: What about Newton.....
J: More newtons per minute
E: Is that what you're teacher taught you?
J.....yeah?
J: I'm not getting her into trouble am I?
E: Not at all
J: Shall I clear up?

The possibility remains that the child misinterpreted the teacher instruction, however an 'intelligent' child has clearly come away from the classroom with Aristotelian type notions. Could an equivalence be drawn here between a child coming away from a classroom with the understanding that 6+7=9? Future research will examine how the design may be improved through enhancement and the incorporation of artificial intelligence.

This research is financially supported by a post-graduate award from The Open University (UK)

Acknowledgements: Research of the kind described in this paper is virtually impossible without a great deal of help and cooperation from several quarters. Firstly, I would like to thank my doctoral supervisor, professor Tim O'Shea. Dr. Ann Jones, Gill Kirkup and Fiona Spensley are gratefully acknowledged for their valuable help in the preparatory stages of this research. Helen Purchase, Engineering Department, University of Cambridge (UK) is also warmly acknowledged for her valuable help in providing initial direction in children, computers and education. Secondly, Mrs. Jacqueline Mundy the school teacher who inspired me with her pedagogical wisdom; Ted Pauley the head teacher, and the staff of St. Mary St. Giles Middle School, Stony Stratford, Milton Keynes (UK) are warmly thanked for their unhindering help and kindly allowing evaluation research to be conducted in their school. Thirdly, and by no means least, the children of the current third year are warmly thanked for their time, help and extremely valuable contributions which are still in continuance.

References

Bento, S.G. (1990) A Holistic model for the evaluation of intelligent tutoring systems. *Interactive Learning International*, 6, 4, 173-180.

Bento, S.G. (1991) Augmenting children's qualitative reasoning processes using a microworld for Newtonian mechanics. Paper submitted to Knowledge Based Environments for Teaching and Learning. Sixth International Conference (May 31-June 2), Genova, Italy.

Cooper, L. (1935) *Aristotle, Gallileo, and the Tower of Pisa*, New York: Cornell University Press.

Darrow, C. (1930) *The New World Physical Discovery*, Indianapolis.

De Kleer, J. and Bobrow, D.G. (1990) Qualitative reasoning with higher order derivatives, in Weld, D.S. and De Kleer (eds.) *Readings in Qualitative Reasoning about Physical Systems,* pp. 127-177, Morgan Kaufman Publishers, California.

De Kleer, J. and Brown, J.S. (1990) A qualitative physics based on confluences, in Weld, D.S. and De Kleer (eds.) *Readings in Qualitative Reasoning about Physical Systems,* pp. 88-125. Morgan Kaufman Publishers, California.

Department of Science (1983) Science in primary schools: a discussion paper produced by HMI Science Committee. HMSO (UK).

Department of Science (1989) *Aspects of primary education: the teaching and learning of science.* HMSO (UK).

diSessa, A.A. (1982) Unlearning Aristotelian physics: a study of knowledge-based learning, *Cognitive Science* 6, 37-75.

Education Reform Act (1988) Chapter 40 HMSO (UK).

Ericsson, K.A. and Simon, H.A. (1984) Protocol analysis: Verbal reports as data. Cambridge, MA: MIT Press.

Forbus, K.D. (1990) Qualitative physics: past, present, and future. *Readings in Qualitative Reasoning about Physical Systems,* in Weld, D.S. and De Kleer (eds.) pp. 11-39, Morgan Kaufman Publishers, California.

Gadamer, H.G. (1988) *Truth and Method (Third Impression).* Sheed and Ward: London.

Jones, A. and Scrimshaw, P. (1988) Exploring the curriculum: Computers in education 5-13. The Open University Press, Milton Keynes.

Kemmis, S. (1976) Evaluation and the Evolution of Knowledge about Educational Programs. Ph. D Thesis, University of Illinois at Urbana-Champaign.

Kuipers, B. (1990) Qualitative simulation, pp. 236-256, in Weld, D.S. and De Kleer (eds.) *Readings in Qualitative Reasoning about Physical Systems*, pp. 236-256 Morgan Kaufman Publishers, California.

Nisbett, R.E. and Wilson, T.M. (1977) Telling More than we can know: verbal reports on mental processes. *Psychological Review*, 84, 3, pp. 231-259.

Pea, R.D. (1987) Integrating Human and Computer Intelligence. In R.D. Pea and K. Sheingold. *Mirrors of Minds: Patterns of Experience in Educational Computing* pp. 128-146.

Scanlon, E. and O'Shea, T. (1986) *Educational Computing*. John Wiley and Sons in association with the Open University.

Science in the National Curriculum (1989) Department of Education and Science and the Welsh Office. HMSO (UK).

Spensley, F.; O'Shea, T; Singer, R; Hennessey, S; O'Malley, C. and Scanlon, E. (1990). An 'Alternate Realities' microworld for horizontal motion. *International Conference on Computer Assisted Learning, Proceedings of 3rd International Conference (ICCAL '90)*, Hagen, 11-13 June 1990. Springer-Verlag: Berlin.

Technology in the National Curriculum (1990) Department of Education and Science and the Welsh Office. HMSO (UK).

Viennot, L. (1979) Spontaneous reasoning in elementary dynamics, *European Journal of Science Education* 1, 203-221.

Winograd, T. and Flores, F. (1988) *Understanding computers and cognition: A new foundation for design*. Ablex Publishing Corporation. Norwood: NJ. pp. 27-30.

An Adaptive ITS Architecture for an Industrial Training Problem

Alberta Bertin, Fabio Buciol, Giovanna Dondossola, Cristina Lanza
CISE Tecnologie Innovative Spa,
Artificial Intelligence Section,
P.O.Box 12081, I - 20134 Milano, Italy

Abstract: This paper describes an Intelligent Training System architecture to support recurrent training for maintenance personnel of electric distribution networks. The main task of the *Electrical Equipment Maintenance Training* (EEMT) system is sensitisizing personnel to safety precautions, especially related to the risks involved while operating on live electrical devices.

Expert operators tend to underestimate risks and often disregard safety precautions. The analysis of their way of acting resulted in the development of a cognitive model of the operator's behaviour for supporting error diagnosis, and in determining effective training methods.

The EEMT is an adaptive simulation-based training system which effectively integrates multiple domain models and multiple training methods supported by different training strategies. Individualized training is achieved by planning and replanning the instructional activity within a session based on the trainee state of knowledge.

Introduction

Training operators of complex processes and sophisticated systems is crucial for those activities which involve personnel security and/or the safeguarding of plants and of the environment. The training process can require several months; furthermore, skilled personnel must undergo recurrent training courses to review knowledge and bring the operational capabilities up to date, so that a considerable amount of resources are invested. Simulators have been widely used to support training, reducing safety problems and training costs. Operators can practice operating procedures in safe conditions and look at the effects of their actions on the simulated process developing their own models of the system behaviour. However, simulators are limited instruments for training practice because they cannot actively support the trainee learning process. Simulators cannot detect and point out trainee errors and do not provide explanations and error remedies.

Artificial Intelligence techniques have been used to develop Intelligent Training Systems (ITSs) that can overcome all these limits. Existing mathematical simulators have been often used as a starting-point to build ITSs, adding knowledge based modules to provide expert functionalities. However, no existing system fully reflects the reference architecture of an ITS, either because basic modules are missing, or show poor functionalities.

Effective graphical interfaces to support interactive inspectable simulations have been added to simulators to help the construction of strong cognitive models of the process to be taught (Hollan et al., 1984; Woolf et al., 1986). Knowledge-based techniques have been applied to maintain explicit representations of operating procedures and heuristics (Woolf et al., 1986; Ritter and Feurzig, 1988), and model based simulations have been used to support explanations about the physical process (Brown et al. 1982; White and Frederiksen, 1986). Multiple tutoring methods and a trainee model have been developed to provide customized training (White and Frederiksen, 1986; Murray, 1989). Anyway, no one integrates multiple domain models, training strategies, didactic methods, and human-computer interaction functionalities, thus supplying a **complete and effective training for a real industrial problem.**

This paper proposes an adaptive architecture for ITSs in industrial environments which integrates all those features highlighting adaptive training and error diagnosis. The proposed architecture has been used to provide Electrical Equipment Maintenance Training. The aim of the system is to support the current training practice focusing on the improvement of awareness of risky situations and conscious application of appropriate safety precautions.

A Reference Architecture for an Industrial Training System

The reference architecture for an ITS generally includes three main subsystems: the **Domain**, the **Tutor** and the **Human Computer Interface**. In designing ITSs for coping with industrial applications we propose to subdivide them identifying constituent subparts:

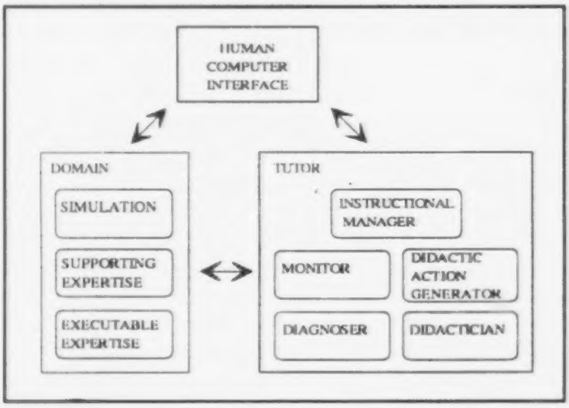


Figure 1. Reference Architecture

In the **Domain** part, all the different types of knowledge required for training is represented. Three main subparts have been identified: *Simulation*, *Supporting Expertise* and *Executable Expertise*.

Simulation contains knowledge for generating the behaviour of the physical system, such as structural and functional models, and inference mechanisms for quantitative and/or qualitative simulations.

Supporting Expertise comprises the structural model of the physical system as it appears to the trainee, including the equipment, the working tools and protective clothing to be used for accomplishing working procedures and supporting the management of the trainee operations.

Executable Expertise includes procedural and associational models representing maintenance procedures and safety rules which specify the precautions to employ while operating. By executing maintenance procedures and safety rules the training system can simulate the expert behaviour.

The **Tutor** includes five main functional modules: the *Instructional Manager*, the *Didactician*, the *Didactic Action Generator*, the *Monitor* and the *Diagnoser*, and a **trainee model** and a set of knowledge bases supporting both the didactic and the diagnostic activities.

The *Instructional Manager* controls the overall training process, co-ordinating the other Tutor modules by sending specific messages in order to adapt the session to the individual trainee learning needs (e.g., providing error remedies to be applied in session re-planning).

The *Didactician* creates and executes a plan for the session to achieve a set of specified training objectives, and can modify this plan to customize training to the specific trainee needs.

The *Didactic Action Generator* generates the appropriate explanation to a trainee error, query or help request. The *Monitor* compares the trainee behaviour with the expert behaviour, detecting discrepances and the *Diagnoser* interprets the trainee error and generates diagnostic hypotheses.

The **Human Computer Interface** implements the interaction specified by the Didactician in the session plan and takes trainee inputs: actions to accomplish a maintenance task, answers to a test or a dialogue, queries from the trainee to the Didactician.

It interfaces the system to the trainee and the human instructor: the *trainee interface* is characterized by textual and graphical presentations, exercises and tests to accomplish the teaching process; the *instructor interface* allows the human trainer to specify the training contents and access the trainee model to verify the acquired performances.

Adaptive Training supported by Multiple Domain Models

A main feature of an ITS is the capability of tailoring instruction to the individual trainee needs. The ability of an ITS of adapting the training process to the incoming learning needs and to the trainee own style is mainly provided by a planning facility guided by an individual trainee model, and by the separation between the content of instruction and the way to teach it (Peachey and McCalla, 1986; Murray, 1989).

The EEMT system has been designed according to the reference architecture previously described. The separation between domain and tutor modules is neat, increasing the system modularity and furthermore, domain knowledge can be easily accessed and used for training purposes. In fact, multiple domain models support the didactic activity contributing to the adaptibility of the whole training system. In the EEMT system several training functionalities are designed to take care of the individual trainee learning needs and preferences, the most of them needing a well-structured domain knowledge representation to couple the suitable way of teaching, specified by the didactician, with a precise knowledge unit.

Adaptive training is pursued by selecting among three different *strategies* which characterize the modes of interaction between the trainee and the system, by *planning* and *replanning* a training session adopting different *methods*, and by giving *explanations* tailored on the trainee knowledge and needs.

Tutoring, *Coaching* and *Self-Teaching* strategies establish the teaching style to be adopted during a session. Trainees are given different degrees of assistance and freedom depending on their needs and preferences. The *Tutoring* strategy particularly suits the needs of inexperienced trainees and intervenes with explanations and suggestions whenever the trainee makes a mistake or has some difficulties. The *Coaching* strategy is addressed to expert trainees showing mastery of the teaching subject and intervenes only when the trainee violates safety rules. The *Self-Teaching* strategy is devoted to those users which are conscious of their knowledge and needs, and can autonomously determine the training session. The EEMT training objectives have been identified according to the Bloom's taxonomy (Bloom, 1956). They specifically pertain to the cognitive area and namely: teaching of technical background *knowledge*, its *application* in maintenance tasks, and *evaluation* of the danger while manipulating electrical equipments. The EEMT system achieves the above objectives by using *problem-driven*, *example-driven*, *explanation-driven* and *assessment* methods. The problem-driven method lets the trainee cope with a training problem *acting* on the simulation of the physical system; in the example-driven method the system *shows* the expert behaviour executing a maintenance task; in the explanation-driven method the system *gives a lecture* on technical background knowledge, and *evaluates* the trainee theoretical knowledge.

The Tutor selects a set of training methods to achieve each objective on the basis of information about their effectiveness towards the trainee. A skeletal plan is defined for each method as a sequence of top level training goals that must be refined by the Didactician depending on individual trainee features. The Didactician is also in charge of executing the *session plan* possibly *replanning* it according to the diagnosis of trainee errors.

Different *levels of explanations* allow to tune the tutorial dialogue, adapting it to the trainee comprehension capabilities and attitudes. Trainee performances and frequency in querying the system about further detailed explanations are significant in choosing the suitable level to be adopted in giving error and query explanations (Nicolosi and Bertin, 1990).

Multiple domain models are provided, supporting the tutoring activity with the precise piece of knowledge the system has to deal with. Referring to the identified training objectives and to the chosen methods, the Domain can effectively provide to the Tutor the kind of knowledge which is necessary to carry out the session, either for teaching domain knowledge and generating explanations, or for assessing the trainee state of knowledge and skills. *Functional*, *structural*, *behavioural*, *procedural* and *associational* models can then be used according to the identified needs, with respect to both training requirements and trainee features.

In the EEMT system the simulation of the physical system is constantly maintained to compute the dagree of danger related to a specific situation and display it on the trainee interface, the main system objective being to sensitize experienced personnel to safety problems when dealing with electrical equipments (Bertin et al., 1991). Any accident situation related to the execution of a wrong procedure or to the violation of safety rules are detected and simulated, explaning both the causes and the effects in terms of equipment damage and injury to the operator.

The Diagnosis of Trainee Errors

Two fundamental tasks of ITSs in industrial environments are monitoring and diagnosing the trainee's behaviour while performing operating procedures. An individualized instruction is possible just on the basis of the evaluation of trainee performances and on a deep analysis of the error causes. The first one allows the selection of the appropriate next exercise and the choice of the most suitable level of explanation, while the latter guides the generation of the error explanation and suggests the remedy to be applied for removing the misconception or filling up the lack of knowledge.

The operating procedure can be seen as a plan which can be defined as:

"a . . . representation of both a goal (together with its intermediate sub-goals) and the possible actions required to achieve it; where both **are related to some time period** which may be stated either very precisely or only in the vaguest terms, but which must be specified in some degree" (Reason, 1979, p.69).

In EEMT a plan is a set of partially ordered goals to performe a maintenance task, each of them can be achieved by a set of correct and incorrect actions.

The operator actions can semantically fail mainly in two different ways: the action may be correct, but the plan inadequate, or the plan may be satisfactory but the action is not correct. Referring to EEMT, operators have to perform maintenance procedures in safe conditions, this requires that each operator action is executed according to the general safety precautions to be applied when acting on live electrical devices. Each goal must be achieved executing an action which will be composed of *working tools*, *protective clothing* and a *component* of the electrical equipment, the appropriate action depending on a set of safety rules. In such a scenario, monitoring trainee's behaviour consists of detecting a discrepancy referring to the expert model generated from the domain knowledge, while diagnosis consists of the interpretation of the trainee error and the identification of the causes which generated it. In the EEMT system the Monitoring has been carried out separately for the goal and for the action. The trainee goal must fit one of the expected goals of the maintenance procedure and the trainee action shall not violate any safety precaution. Thus, **monitoring the trainee goal** comes out with an assessment of the trainee performance which recognizes *optimal* goals, *acceptable*, *wrong* and *illegal* ones. Wrong and illegal goals are related to the execution of an inadequate plan, where the wrong ones correspond to errors related to the plan execution and the illegal ones implies the violation of safety criteria.

Monitoring the trainee's actions depends on the previous evaluation of the corresponding goal. For goals related to the execution of an inadequate plan, the action to achieve it can be either an *unsafe action*, when it violates some safety rules, or an *irrelevant action* when it is safe but useless to the plan execution. For goals related to a satisfactory plan the action to achieve it can be either *unsafe* or *safe*.

Diagnosis is activated anytime the monitoring process detects a discrepancy. The above definition of plan helped us in the classification of error types for the EEMT domain. The interpretation of trainee errors is carried out classifying the detected discrepancy for the goal and for the action basing on the error types described in a *bugs catalogue*.

There are four possible error types for inadequate plans: **intrusion** (the trainee achieved a goal which is not part of the plan for the maintenance task), **omission** (a goal of plan has been omitted), **reversal** (two goals of the plan have been reversed), **misplacement** (a goal of the plan has been achieved at the wrong time). There are three error types for improper actions: **intrusion** (e.g., some irrelevant working tool has been used), **omission** (e.g., some protective clothing have been omitted), **replacement** (e.g., the proper working tool has been replaced with an improper one).

Next, after the error classification, the generation of hypotheses concerning the mental causes of the bug is activated. A classificatory reasoning about symptoms is used (Johnson and Soloway, 1984) basing on a cognitive model of the operator errors. Diagnostic knowledge is represented by classificatory rules relating symptoms to disease classes. The condition part of the rules checks domain dependent parameters and trainee's working experience (e.g., the complexity of the procedure, the frequency of application of the procedure for the trainee); the action part of the rules provides a set of diagnostic hypotheses in terms of **lack of knowledge or skills**, cognitive problems derived from doing similar or more frequent tasks, such as **matching-bias** and **frequency-gambling**, **misconceptions**, **lack of motivation**, **slips**, loss of temporal ordering among steps of the procedure, such as **lapses** (Reason, 1987). A degree of confidence is associated to each diagnostic hypothesis taking into account the operator's training history. The most probable hypotesis is then considered by the Tutor in order to provide explanations, to identify the error remedy and to replan the training session.

Acknowledgements: The work described in this paper has been undertaken in ESPRIT Project P2615 - ITSIE. The project is partly supported by the C.E.C. under the ESPRIT programme, and by the Automation Research Center of the E.N.E.L., the Electricity Board of Italy.

ITSIE was developed by a consortium composed of the following partners: Marconi Simulation, CISE, Iberduero, (with support from Labein and the Basque Country University), Alcatel Alsthom Recherche, CRI, Heriot Watt University. The authors wish to acknowledge the contribution of all the members of the project team to the work and in particular to the specification of the ITSIE reference architecture presented in section 2 of this paper.

References

Bertin, A., Buciol, F., Dondossola, G. and Lanza, C. (1991). *An ITS for high risk tasks training.* Proceedings PEG '91 - Knowledge Based Environments for Teaching and Learning. Rapallo.

Bloom, B.S. (1956). *Taxonomy of Educational Objectives: Book1. Cognitive Domain*. New York: Longmans Green.

Hollan, J.D., Hutchins, E.L., Weitzman, L. (1984). STEAMER: An Interactive Inspectable Simulation-Based Training System. *AI Magazine* 5: 15-27.

Kearsley, G. (ed.) (1987). *Artificial Intelligence and Instruction*. Addison Wesley Publishing Company.

Murray, W.R. (1989). *Control for Intelligent Tutoring Systems: A Blackboard-based Dynamic Instructional Planner.* Proceedings of the 4th International Conference on AI and Education. Amsterdam: IOS.

Nicolosi, E., Bertin, A. (1990). *An Analysis of Explanation Types in Intelligent Tutoring Systems*. Proceedings of the 5th Workshop on Explanation. Manchester.

Peachey, D.R. and McCalla, G.I. (1986). Using planning techniques in intelligent tutoring systems. *Int. J. Man-Machine Studies* 24: 77-9.

Reason, J. (1979). Actions not as plan: The price of automatization. In J. Underwood and R. Stevens (eds.) *Aspects of consciousness*, Vol. I. Psychological issues. London: Academic Press.

Reason, J. (1987). Generic Error-Modelling System (GEMS): A Cognitive Framework for Locating Common Human Error Forms. In J. Rasmussen, K. Duncan and J. Leplat (eds.) *New Technology and Human Error*. John Wiley & Sons Ltd.

Ritter, F., Feurzig, W. (1988). Teaching Real-time Tactical Thinking. In Psotka, J., Massey, L.D. and Mutter S.A. (eds.) *Intelligent Tutoring Systems: lessons learned*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Sleeman, D. and Brown, J.S. (eds.) (1982). *Intelligent Tutoring Systems*. London: Academic Press Inc.

White, B.Y., Frederiksen, J.R. (1986). Intelligent Tutoring Systems Based upon Qualitative Model Evolutions. Proceeding of the 5th National Conference on Artificial Intelligence. Philadelphia.

Woolf, B., Blagen, D., Jansen, J.H., Verloop, A. (1986). *Teaching a Complex Industrial Process*. Proceedings AAAI-86. Philadelphia: Morgan Kaufmann Publishers Inc.

Learning to Learn: The Implications of Strategy Instruction in Computer Programming

Katerine Bielaczyc and Margaret M. Recker
Graduate School of Education
University of California, Berkeley 94720

Abstract: This research investigates the relationship between the self-explanations students generate while learning from instructional materials and their subsequent problem-solving and cognitive skill acquisition. Previous research in self-explanations found that good-performance students not only generated more self-explanations, their self-explanations were qualitatively different. Specifically, the strategic aspect of self-explanations seems to play an important role in the learning process. The present paper discusses three lines of research directed toward examining this role more closely: experimental investigations and analyses, cognitive modelling, and the design of instructional environments. These lines of research are interdependent, and iterate and inform one another in important ways. The research provides insight into the complex interactions between learners' knowledge, explanation strategies, and instructional materials, and works toward a deeper understanding of learning in complex problem-solving domains such as computer programming.

Introduction

In recent years, there has been an increasing amount of research examining learners as they are studying instructional texts and examples prior to problem-solving. This area of research is a natural extension from the previous decade's focus on problem-solving, skill acquisition, and expertise. That is, we see a broader focus from investigating how problem-solving skills are used and developed to the earlier phase of learning when the concepts and principles of a domain are initially encoded. Investigations examining these initial phases have tended to either manipulate the style of the instruction and presentation of subject matter (e.g., Charney & Reder, 1986) or characterize the individual differences in students' learning strategies on a given set of instructional materials (e.g., Chi et al., 1989; Ferguson-Hessler & de Jong, 1990). These manipulations or differences are then viewed in terms of their impact on or relation to subsequent problem-solving tasks.

One particular learning strategy that has been examined is that of *self-explanation*. This refers to the process by which learners elaborate on the texts and examples as they are reading and learning from instructional materials. Our interest is in how learners encode and use their interpretations of text and examples in learning to write computer programs. Previous research in physics (Chi et al., 1989) and LISP programming (Pirolli and Bielaczyc, 1989) found that particular characteristics of the self-explanations students made while studying instructional materials correlated with the corresponding problem-solving performance of those students. Good performance students not only generated more self-explanations than poor performers, their explanations were qualitatively different. Good performance students also seemed better able to detect their own comprehension failures and showed other effective metacognition. In the domain of programming, three characteristically good self-explanation strategies were identified:

- identifying and relating the main ideas of concepts discussed in the text,
- determining both the form and meaning of the LISP code presented in the examples, and
- connecting the ideas in the texts with their instantiations in the examples, and vice-versa.

Learner self-explanation strategies have been the subject of further examination by Pirolli and Recker (1991). The work suggests that the strategic aspect of self-explanation is an important component of the learning process. The present paper reports on three lines of research directed toward exploring this component in more depth: (1) experimental investigations and analyses of self-explanation strategies, (2) the development of a cognitive model of strategy use in self-explanation and (3) a preliminary instructional environment designed to promote and encourage effective self-explanations.

Elaborations Studies and Analyses

One approach toward examining the effectiveness of self-explanation strategies has been through experimental investigations and more detailed analyses of the content and structure of learners' self-explanations. We recently conducted a study investigating the effects of training students in self-explanation strategies on learning to program (Bielaczyc, Pirolli, & Brown, 1991). In the earlier self-explanations research, the relationships between self-explanation strategies and performance were identified through correlational rather than experimental analyses (Chi et al., 1989; Ferguson-Hessler & de Jong, 1990; Pirolli & Bielaczyc, 1989; Pirolli & Recker; 1991). Our training study had two main aims, (a) methodological–to gain a better sense of the causal role of self-explanation strategies; and (b) pedagogical–to examine whether such strategies could be effectively taught.

A set of instructional interventions based on modelling and scaffolding techniques was used to communicate the characteristically good self-explanation and metacognitive strategies identified by our previous research. These training interventions took place in a longer sequence of instructional lessons on programming using the CMU LISP Tutor (Reiser, Anderson, & Farrell, 1985). The instructional lessons typically involved learning from expository materials presenting relevant examples, and then working through associated programming exercises.

The results of the study reflect both quantitative and qualitative improvements in the self-explanation and monitoring strategies of subjects receiving strategy training. The subjects who received strategy training also showed significantly greater performance gains on associated programming problems than subjects in an instructional control group. There was an accompanying improvement in the acquisition of programming skills with improved self-explanation strategies. These results strengthen the earlier correlational findings by providing an experimental demonstration that the self-explanation strategies are related to more effective learning.

Analyses of the structure and content of the self-explanations students made while reading the instructional materials provide characterizations of both effective general and individual learning strategies. The work suggests substantive and interesting differences in the types of self-explanation and metacognitive strategies used by individual learners. It also indicates the need for further examination of the learning which occurs prior to problem-solving.

Implementation of a Cognitive Model

Elsewhere we described a preliminary formalization and partial implementation of the self-explanation processes in a Soar computational model (c.f. Pirolli & Recker, 1991). Soar is a general problem-solving architecture which includes an experience-based learning mechanism, called chunking (Newell, 1990).

For the current research, a model was implemented which focussed only on the three strategies described above to process instructional material.. In this implementation, the model has two main problem spaces, explain-text and explain-example. Each of these problem spaces contains operators which correspond to the strategies taught in the strategy training research. Application of the strategy operators results in a set of chunks which capture important aspects of the instruction. These chunks might then be used in resolving impasses that may arise during subsequent programming.

For example, in the explain-example problem space, the example is encoded as a set of linked lists, where each list represents a piece of code in the example. This problem space has knowledge of three operators: *find-form, find-meaning, and make-ties*. The *find-form* operator identifies the piece of Lisp code to process, the *find-meaning* operator retrieves relevant Lisp knowledge about the function of that piece of code, and the *make-ties* operator attempts to relate the piece of code to previously read instructional text. The system processes the example by applying, in succession, each operator on each sub-list. Each operator application adds information to an emerging explanation structure, resulting in an embellished and integrated structure.

The cognitive model, through the application of strategic operators, constructs chunks of integrated knowledge. Removing certain operators will result in less or sparser chunks, which may have an impact during subsequent problem-solving impasses. Particular deletions may correspond to variations observed among poorer performing subjects in the empirical studies.

However, this simple model does not account for the full range of behaviors exhibited in the protocols. For example, the model does not explain differences in metacognitive abilities among subjects. It also does not explain how subjects detect inconsistencies in their explanations nor how they decide when a particular self-explanation is satisfactory. These represent modelling issues to be addressed.

Implications for an Instructional Environment

Research on strategy use in self-explanation has many implications for the design of instructional materials. Toward this end, we have designed a computer-based instructional environment that presents instructions on programming recursive functions in Lisp. The environment, embedded in a hypertext environment, presents textual information in a hierarchy from general to specific, integrates examples within text, and presents instruction on an abstract model of how recursive programs are coded. The structure of the environment is intended to promote and encourage the construction of effective self-explanations.

The environment provides support for the self-explanation strategies listed above in the following manner. First, it allows learners to focus on the main points of the instruction by structuring the instruction in a hierarchy from general to specific. The learner controls the viewing of additional, more specific, textual elaborations. Second, it provides examples containing elaborations which can help the learner determine the meaning of code fragments. These elaborations are initially hidden and become visible only if requested by the learner. Also, these elaborations explain the goals, conditions, and situations that apply to each particular code fragment and thus allow the learner to connect the example code to the main points presented in the text.

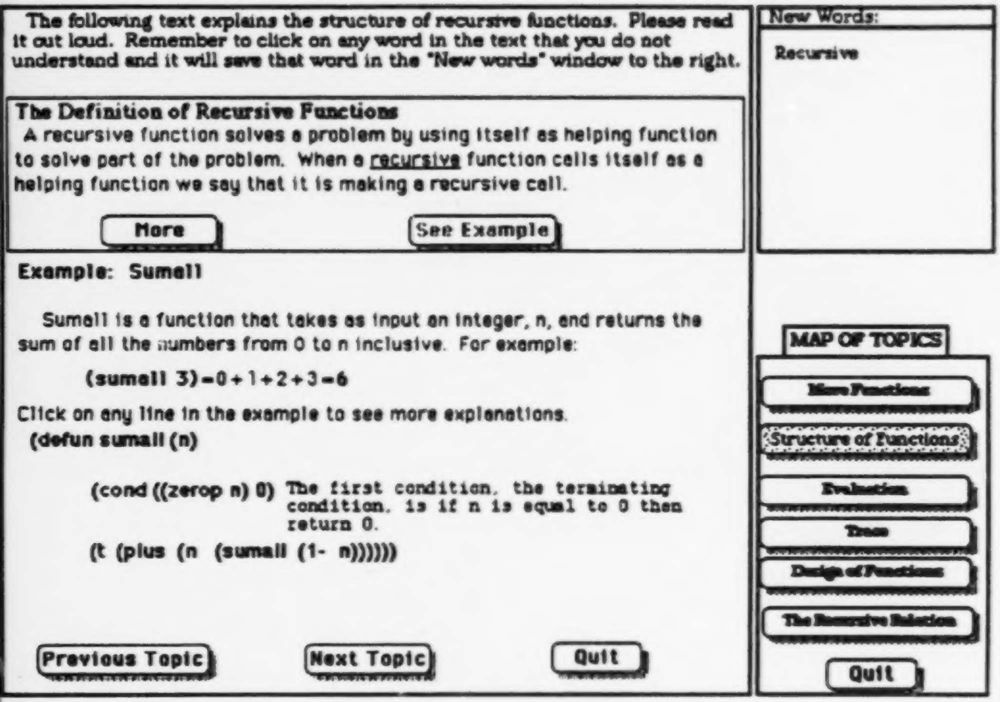
Finally, the environment offers support for the kinds of metacognitive reasoning that was shown to be important in the self-explanation research. Learners in this environment are provided with the ability of highlighting terms in the instruction that they do not understand. The highlighting feature can provide implicit monitoring of one's own state of comprehension and mark potential explanation goals. In addition, at any time, learners can select a word that they have highlighted and type in and save their own definitions for that word. This feature was motivated by findings showing the general superiority of self-generated elaborations over text-supplied elaborations (Reder, Charney, & Morgan, 1986).

A sample interaction from the instructional environment is presented in Figure 1. The learner is viewing instruction on the "structure of recursive functions." The main point of the instruction is displayed in the box near the top of the screen. In this text, the learner has chosen to highlight the word "recursive", which caused the word to be underlined and to be displayed in the "New Words" window at the upper right hand corner of the screen. The learner has also selected the "See Example" button which caused an example recursive function to be displayed in the bottom half of the screen. Additionally, the learner has requested an explanatory elaboration about the second line of the example. This elaboration is displayed next to the Lisp code. Also note the "Map" at the lower right hand side of the screen. This map presents an overview of the topics contained in the recursion lesson and thus provides a global scheme for navigating through the instruction. The buttons located at the bottom of the screen allow the user to review previous parts of the lesson or to continue on through the instructional topics.

We are currently conducting an empirical study to assess the impact of the explanation environment on students' learning. The basic design of the study involves contrasting the self-explanations that subjects generate while using the instructional environment to those generated by subjects using a second environment which was specifically designed to reflect more standard, linear instruction (c.f. Recker, 1991).

Weaving of the Threads

The three lines of research described in the present paper provide insight into the strategic aspect of self-explanations. The results of the empirical studies point to the importance of students' strategies during the initial stages of learning and their implications for programming skill acquisition. Additionally, the training study showed that pedagogical manipulations can help students to improve their learning strategies. The implementation of the cognitive model provides a formalization that captures the strategic aspect of self-explanations. And finally, the instructional environment shows how results from psychological research can be used to design educational systems, an important application of cognitive science research.



The following text explains the structure of recursive functions. Please read it out loud. Remember to click on any word in the text that you do not understand and it will save that word in the "New words" window to the right.

The Definition of Recursive Functions

A recursive function solves a problem by using itself as helping function to solve part of the problem. When a recursive function calls itself as a helping function we say that it is making a recursive call.

Example: Sumall

Sumall is a function that takes as input an integer, n, and returns the sum of all the numbers from 0 to n inclusive. For example:

```
(sumall 3)=0+1+2+3=6
```

Click on any line in the example to see more explanations.

```
(defun sumall (n)

     (cond ((zerop n) 0) The first condition, the terminating
                         condition, is if n is equal to 0 then
                         return 0.
     (t (plus (n  (sumall (1- n))))))
```

Figure 1.

However, these three lines are not isolated, rather they influence and inform one another in significant ways. The experimental studies identify cognitive factors that are involved in effective self-explanations and that need to be accounted for in a cognitive model of the process. The studies also indicate aspects of learning that should be offered support in an instructional environment.

The implementation of a cognitive model provides a concrete formalization of the empirical data and an instantiation of the interpretation of the results. The implementation of the model may also uncover assumptions which need to be strengthened through further empirical investigations. The cognitive model additionally provides constraints on decisions made in the course of designing the instructional environment. Ultimately, a cognitive model could form the basis of a student modelling component that is sensitive to individual differences among learners' strategies. For example, it could be used to provide advice on effective self-explanations or to explicitly model helpful strategies to learners.

The development of an instructional environment not only draws on information provided by the empirical studies and cognitive model, but, as above, its design may reveal areas that require further support from research and modelling. The instructional environment also provides a tool for observing students' learning in great detail. Such data may contribute to and increase our understanding of how students come to learn from instructional materials.

Conclusion

The research discussed here examines aspects of the strategies students use to explain instructional materials to themselves prior to problem-solving. The research was framed by considering three related areas of research. The first involved detailed protocol analyses and a training study concerning student learning strategies. The second focussed on the implementation of a cognitive model. The third discussed the design of an instructional environment.

The approach taken here highlights the complex interactions between learners' knowledge, explanation strategies, and instructional materials. We propose that there are important and interesting individual differences among the kinds of learning strategies used by learners to encode knowledge in complex problem-solving domains. This poses a challenge to instructional design and student modelling efforts by emphasizing the need to take individual variability into account.

Acknowledgements: Portions of this research were funded by the National Science Foundation (award number IRI–9001233) to Peter Pirolli, Evelyn Lois Corey Research Fund and NICHD (NICHD-06864) to Ann Brown, and a grant from the C.D. Weinman fund in instructional technology to the second author. We thank Peter Pirolli, Elliot Soloway, and Michael Reynolds for comments on an earlier draft of this paper.

References

Bielaczyc, K., Pirolli, P., and Brown, A. (1991). *The effects of training in explanation strategies on the acquisition of programming skills*. Paper presented at the Annual Meeting of the American Educational Research Association, Chicago, Illinois.

Charney, D. and Reder, L. (1986). Designing interactive tutorials for computer users. *Human-Computer Interaction*, *2*, 297-317.

Chi, M.T.H., Bassok, M., Lewis, M.W., Reiman, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science*, *13*, 145-182.

Ferguson-Hessler, M.G.M. and de Jong, T. (1990). Studying physics texts: Differences in study processes between good and poor solvers. *Cognition and Instruction*, 7, 41-54.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Pirolli, P. and Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ: Erlbaum.

Pirolli, P. and Recker, M. (1991). *Knowledge construction and transfer using an intelligent tutoring system: The role of examples, self-explanation, practice, and reflection*. Technical Report CSM-1, University of California, Berkeley.

Recker, M. (1991). *Learning recursion through the construction of explanations*. Paper presented at the Annual Meeting of the American Educational Research Association, Chicago, Illinois.

Reder, L., Charney, D., and Morgan, K.(1986). The role of elaborations in learning a skill from instructional text. *Memory and Cognition*, 14:64-78.

Reiser, B.J., Anderson, J.R., and Farrell, R.G. (1985). Dynamic student modelling in an intelligent tutor for Lisp programming. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 8-14*. Los Altos, CA: Morgan-Kaufman.

Teaching by Means of Social Simulation

Eli Blevis and Alex Kass
Institute for the Learning Sciences and
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, Illinois 60201

Abstract: An effective way to teach social skills is to allow a student to interact with a social simulation program. In this paper we describe one such program, which we are building. We argue for certain interface techniques, and describe our methods for building realistic agents. We describe our approach to providing guidance and feedback to the student, both within the simulated world of the program, and by means of tutoring features outside of the simulation.

Introduction

When you want to learn how to perform a new kind of task, no amount of abstract advice can substitute for trying it yourself. It is only through performing the task that you can see first hand what causes success, and what causes failure. This is generally a very effective way to learn a new skill unless the cost of failure is prohibitive. That's why the flight simulator is such a great teaching tool; it provides most of the advantages of actually flying a plane, while allowing the learner to make critical mistakes without incurring the costs that result from making those mistakes in real life.

The idea underlying teaching by means of social simulation is to accomplish for a social environment what the flight simulator accomplishes for the physical environment of the cockpit. Social simulations teach how to perform a task that involves social interaction by allowing the student to interact with a set of simulated agents in a simulated environment.

Social simulations are particularly appropriate for teaching the skills required in domains such as diplomacy, negotiation, and business. For instance, inexperienced business consultants need to learn the delicate skills required to successfully interact with client organizations. Such skills include:

- discovering the official and unofficial structures of client organizations;
- dealing effectively with different personality types within those organizations, and extracting from people the information necessary to make recommendations;
- and, making appropriate recommendations in a style that is convincing but not threatening.

These skills are very hard to teach abstractly. The standard way for consultants to learn them is through experience, but errors that a consultant makes with clients can be quite costly. To solve this problem, we are building a program that simulates the experience of interacting with clients. The program allows the consultant to practice with simulated clients rather than real ones.

In this paper, we will describe how we address some of the many issues one must face when building a social simulation, the following in particular:

- **Constructing effective interfaces:** How can the student interact with the simulated agents? What information needs to be communicated to the user, and how can it be effectively communicated through graphics, text, and video?
- **Building realistic agents:** How does one build a realistic simulation without solving all the problems of AI?
- **Guiding and giving feedback:** Feedback can take place within the simulated world (e.g., the client becomes angry) or outside it (e.g., a teacher intervenes to give advice). Our system employs both types.

In some ways simulations can be even better learning environments than the real world. The simulated environment can be easily manipulated in a way that the real world cannot be; for example, the simulatied environment can be simplified when the student is first learning a skill, and then made more complex as the student improves. Through simulation we can introduce the trainee to a wide variety of scenarios and cultures, letting the student see how his or her actions effect different environments differently.

Interface Issues

The two questions one needs to answer when building a social simulation program are, "How can the student talk to the simulated agents," and, "How is the state of the simulation communicated to the user."

How the Student Communicates with the Program

One of the big questions one must address when designing a social simulation is how the user will perform actions, and communicate with the simulated agents. Ideally, we'd like to allow the student to speak to the agents, or at least to type natural language. However, it is not likely that the problem of parsing unrestricted natural language is going to be solved in the near future. In order to avoid building up false expectations on the part of the user, we need a mechanism for restricting the the things the user can say. Multiple choice input is simple to implement, but there are crucial pedagogical problems with that approach. The principal problem is that presenting the student with a small set of choices gives too much away. In real life, the student will have to be able to formulate responses; the options won't be laid out in advance.

Our system allows the student to construct actions out of a set of abstract vocabulary items. The student first chooses an action, then an object class, and then a specific object from that class. For example, suppose that the student is in a client's office and wishes to ask the client how often stock-outs occur. To accomplish this, the student would first select the action, ASK-ABOUT, then the object-class, PROBLEM-FREQUENCY, and then the object, STOCK-OUTS. Selecting the ASK-ABOUT action restricts the set of object classes to those which are appropriate for that action, and similarly, selecting PROBLEM-FREQUENCY restricts the set of objects.

Having selected what to say, the student is then allowed to decide what tone of voice to use (friendly, formal, or somewhere in between). The system then shows the user the

English sentence that corresponds to what has been chosen, and the user can decide to actually say it, or to construct another statement instead.

This approach has the advantage of forcing the student to construct, rather than select what to say, while restricting the domain of what can be said to things that the agents in the system can understand. In addition, this approach allows us to reinforce a taxonomy of actions, object classes, and objects that we would like a student operating in the simulated environment to learn.

How the Program Communicates with the Student

The program uses several methods of communicating information to the user.

Drawings and Photos: A large portion of the screen is devoted to graphics that show the student what he or she would see in the simulated situation. For instance, when entering an office, the user first sees a drawing of what the room looks like. The way an office looks often conveys important information about its occupant. If the student see's that the client to be interviewed has a corner office, this conveys something about the occupant's status. If the office is messy, this conveys something about the occupant's personality.

In addition, when the student decides to read memos or background documentation, the program presents a hi-fidelity graphical representation of what the actual document would look like.

When communicating with an agent, the student sees a scanned-in photograph of the agent. This lets the student know , for example, the sex of the agent, and how the agent dresses. At some time in the future, we will provide full-motion video instead of still photos.

Gauges: Talking to a still photograph is different from talking to a living, moving person. Perhaps the most important difference is that with a real person, shifting body language and change of facial expression communicates important information about the agent's reactions to events. To make up for this deficiency, the program displays three gauges along with the photograph of the agent. The gauges represent dimensions of the visible state of the agent, specifically how happy/angry the agent looks, how tense/relaxed the agent looks, and how interested/distracted the agent looks. These gauges are not windows into the agent's head; an agent may look happy while actually being very angry; that's one of the things a student has to learn. The gauges merely represent some of the information that, in real life, would be represented by body language and facial expressions.

Text and Sound: When the agents speak to the user, the output is presented as text. This is not the best part of the system. In the future we may use synthesized sound, if we can get it to be high-enough quality.

Building Realistic Agents

The simulated clients and other agents who interact with the user must appear to comprehend and react to the user's language, display moods and emotions, react to observations, and have agendae and expectations that are true to life. Each agent in the simulation needs to be able to take realistic actions and react intelligently to observations. In order to accomplish this, the program represents the states of mind of agents and the process by which agents change their states of mind and take actions. The agents act according to goals, but they do not form complex plans. Their behavior is can be roughly characterized as situation-directed execution in the sense of Firby (1989) and Agre & Chapman (1987).

An agent's state of mind is represented in terms of the following features:

OBSERVATIONS represent what an agent sees or hears. For example, an agent may have observed the user enter the agent's office, or say "Hello."

MOODS represent how an agent feels about another agent. For example, an agent might have a mood of compliance with one agent in the simulation and a mood of alienation towards another.

GOALS respresent an agent's desires to perform actions or achieve a state of the world. For example, an agent may acquire a short term goal to respond to a question from another agent, or a long term goal to bring about the demise of yet another agent.

EXPECTATIONS represent an agent's beliefs about other agents or aspects of the simulated world. For example, an agent may expect another agent to return a courtesy, know about something, or show up for an appointment.

We plan to extend these notions of what are appropriate features of a state of mind to include aspects of various theories in the psychology literature, such as Digman's work on personalities (Digman, 1990).

For each individual, the current state of mind is organized in terms of the features that relate to every other individual known to him or her in the simulated world. For example, the representation of the state of mind for an agent, "Sam," at a point in a simulation could include the following features:

Sam's state of mind with respect to Joey, ordered from foremost to least important is as follows: Sam's MOOD towards Joey is one of annoyance; Sam EXPECTS that Joey will disagree with the need for modernization of the company; bringing about the demise of Joey is within Sam's AGENDA.

Sam's state of mind with respect to the student consultant (user) is as follows: Sam has just OBSERVED the student consultant ask about who would disagree with modernizing the operations of the company; Sam's MOOD towards the user is one of compliance; Sam EXPECTS that the user knows about Sam's opinion that modernization is needed; Sam EXPECTS to discover if the user can be trusted with sensitive information about conflicts; Sam EXPECTS that the user knows that someone at the company might disagree.

At each clock tick in the simulation, each agent decides whether any action is appropriate, based on his or her state of mind and any new observations about the simulated world. Each agent has many decision sturctures denoting potential decisions to act or react to particular observations in particular states of mind. An agent chooses the structure whose activation conditions most closely match his or her current state of mind, and performs the action specified by the chosen structure. In addition, the structure may also specify changes to the agent's state of mind or even changes to that agent's facial expression.

The agent from the previous example, Sam, might have the following two structures from which to choose:

Structure 1:

IF I observe someone ask me who would disagree with me about something
AND LIKELY IF I trust that someone
AND LIKELY IF I need that soneone's alliance
AND IF I know who would disagree with me
THEN I would tell that someone the identity of the person who disagrees
AND THEN I would expect the someone to show sympathy to my side

Structure 2:

IF I observe someone ask me who would disagree with me about something
AND LIKELY IF I don't trust that someone yet
THEN I would tell the someone that I don't konw who would disagree with me
AND THEN I would expect that someone not to ask me again
AND THEN I would show some anger towards that someone

The above does not really capture the full complexity of the decision process we have implemented, but will have to suffice for this brief paper.

Mechanisms of Guidance and Feedback

A simulation-based system can provide a student with two categories of guidance: feedback from within the simulation, and feedback from outside the simulation.

Feedback Within the Simulation

Like the pilot using a flight simulator, the student reveives a lot of feedback simply by using the simulation and observing the effects of his or her actions. For instance, the student's actions may result in getting the needed information, or it may result in evasive, or even hostile action by clients. When the student does not know what the correct thing to do is, he or she finds this out by seeing the results of incorrect actions. Sometimes, the feedback the student receives within the simulation can be more explicit. For instance, the student's manager or colleagues may offer advice or criticism.

The simulation can provide feedback without the intervention an explicit teaching module. However, a teaching module which observes the student's interaction with the simulation and can greatly enhance the quality of feedback, just the way an experienced pilot looking over a student pilots shoulder could enhance the experience of using a flight simulator. The teaching module can enhance the within-simulation feedback by adjusting the simulation to ensure that the student is confronted with situations that will challenge him or her. For instance, if the teaching module observes that the student frequently employs a strategy that would have bad consequences when interacting with a person in a particular mood, it can alter the mood of one of the simulated agents to ensure that the student learns about that negative interaction.

Both in the simulated-world and the natural world, these types of guidance and feedback may be of varying quality and utility. The visible reactions of the simulated client agents may be misleading or difficult for the student to interpret. The casual advice may be useful, or it may be useless. Some people can be trusted, while others cannot. One of the most important things a student needs to learn is to sift through masses of feedback, and to determine what is truly pertinent.

Feedback Outside the Simulation

Sometimes it is worthwhile to give external feedback in a form that is outside the scope of the simulation. Just as an experienced pilot watching a student using the flight simulator might help by occasionally saysing, "pull up," the teaching module can occasionally provide useful commentary or coaching as it observes the student interact with the social simulator. Our system actually contains several, independently operating commentators and coaches, each of which provides a different form of feedback:

Case Presenter: Sometimes the best way to teach a general principle is to tell a story about situation that illustrates the principle. The program contains a large library of case studies, organized and retrieved according to the theory to be presented in Burke (1992). As the program observes the student interact with the simulation, it constantly searches its cases library. When it is "reminded" of a relevant story to tell, it tells it, emphasizing the moral of the story. Of course, the student must still decide how the moral applies to the current simulation.

Video Dramatizations: These are very similar to the stories described above, except they are dramatizations (stored on video disc) of the possible effects of certain actions that consultants can perform.

Open Question Trainer: This component essentially helps the student keep the appropriate goals in mind as the simulation progresses. In the case of business consultants making early contact with a client, the agenda consists of the open questions about the client's operation that the consultant is investigating. When a client says something relevant to one of the current open questions, the system alerts the student that an important question may have been answered, and that answer may have raised important new questions. At first, the system points out questions and answers. As the student becomes more skilled, the program begins to leave it to the student to notice when questions have been answered, and eventually leaves it to the student to raise the questions himself or herself.

Political Analyzer: An important skill that the user must learn is how to extract abstract generalizations from the actions and statements of others. An important class of abstractions that must be recognized are political ones. For instance, the student should be able to recognize statements as threats, or offers of alliance. The political analyzer uses the theories developed by Hughes (1992) to offer political analysis of the dialogs that the student has with the relevant agents.

The external guidance and feedback complement the feedback that comes within the simulation. Without the benefit of the simulated environment, the stories, advice and analysis these modules perform would lack the context necessary to motivate and situate them. When interleaved into the simulation, they are delivered just at the point when they are relevant to what the student is trying to accomplish.

Conclusion and Future Directions

The program we are writing to teach consulting skills by means of social simulation has taught us a great deal about how to construct effective interface features, how to build effective simulated agents, and mechanisms for guidance and feedback. We hope to further refine and develop this promising methodology for educational-software design.

We have also begun to construct tools that allow new structures for agents to be added by a non-programming instructional designer while stepping through the program. These tools automate to some degree the specification of a structure by keeping track of the actual features of an agent's state of mind at points in the program. We consider the further development of these simulation-construction tools to be an important direction for future research.

Acknowledgements: For extensive advice and assistance with this research, including much of the implementation and design work, we are indebted to Robin Burke, Chip Cleary, Marita Decker, Lucian Hughes, Scott Rose, Roger Schank, Chris Sorensen, Tom Murray, and Mary Williamson.

This work is supported in part by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058 and the Office of Naval Research under contract N00014-90-J-4117, by the Office of Naval Research under contract N00014-J-1987, and by the Air Force Office of Scientific Research under contract AFOSR-89-0493. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner.

References

Agre, P.E., and Chapman, D. (1987). Pengi: An implementation of a theory of activity. *Proceedings of AAAI-87*. 268-272.

Burke, R. (1992). *Integrated story retrieval in a case-presentation tutor.* Ph.D. Dissertation. Northwestern University. To appear.

de Waal, F., (1982). *Chimpanzee politics.* London: Jonathon Cape.

Digman, J.M., (1990). Personality structure: Emergence of the five-factor model. *Annu. Rev. Psychol.* 41:417-40.

Firby, J.R. (1989). *Adaptive execution in complex dynamic worlds.* Technical Report YALEU/CSD/RR #672, Yale University.

Goodall, J., (1986). *The chinpanzees of Gombe: Patterns of behavior.* Cambridge, Mass.: Belknap Press, Harvard University Press.

Hughes, L., (1992). *Chimp world: Building social simulations.* Ph.D. Dissertation. Yale University. To appear.

Teaching According to GARP

Bert Bredeweg
Department of Social Science Informatics (S.W.I.)
University of Amsterdam, Roetersstraat 15, 1018 WB Amsterdam, The Netherlands

Radboud Winkels
Department of Computer Science and Law
University of Amsterdam, Kloveniersburgwal 72, 1012 CZ Amsterdam, The Netherlands

Abstract: A growing number of learning environments on the computer is simulation based. Research shows that such simulations are only effective when the student is guided by a tutor, for instance embedded in an Intelligent Tutoring System (ITS). However, most of the simulations are based on complex mathematical procedures. This makes it hard, if not impossible, for the ITS to use the simulator for explanations or Student Modelling, because it has no means to relate the calculations within the simulation to the conceptual framework that represents the knowledge the student possesses, or has to acquire.

An alternative is to use causal, qualitative models as the basis for simulation. We present an integrated qualitative reasoning approach, implemented as a shell named GARP, that can be used to build both domain and student models. The modelling language (at the conceptual level) appears to map directly on that of human reasoning about the same domain. Furthermore, the resulting models in GARP are very suitable for different kinds of explanations, for diagnosis, and for generating critical tests to induce learning or awareness of misconceptions on the part of the student.

Introduction: Simulation in Learning Environments

A growing number of learning environments on the computer is simulation based (see for instance: de Jong (ed.), 1991 for an overview). Usually these environments are based on the "discovery learning" educational philosophy (e.g., Bruner, 1961) and are aptly called discovery worlds. But research shows that such simulations are only effective when the student's actions are monitored by a teacher (human or computer based) and guidance is provided (e.g., Lavoie and Good, 1988; Elsom-Cook, 1990). One solution is to embed the simulation in an Intelligent Tutoring System (ITS). However, most of the simulations are based on complex mathematical procedures that calculate how the specific aspects within the simulation are to be manipulated (e.g., SOPHIE I/II, Brown et al. 1982; STEAMER, Hollan et al. 1984; RBT, Woolf and Cunningham, 1987). These mathematical procedures, although quite efficient in simulating the physical world, provide no conceptual access to the objects and their behaviour in the simulation. This flaw makes it hard, if not impossible, for the ITS to use the simulator for explanations or Student Modelling, because it has no means to relate the mathematical calculations within the simulation to the conceptual framework that represents the knowledge the student possesses, or has to acquire. It is impossible to derive causal explanations of the behaviour of the particular device or system from the mathematical model, so these would have to be added "by hand." Therefore we are looking for different models to use for simulations, models that explicitly represent the objects and their behaviours that play a role in the system. These models should enable causal reasoning, and preferably comply with the way humans reason about the system, i.e., *causal* or *qualitative* models.

Qualitative Reasoning: An AI Perspective

During the last decade qualitative reasoning has evolved as an interesting AI research area that is concerned with automated reasoning about the physical world using qualitative representations. Based on different ontologies about how to represent the physical world in qualitative terms, three basic approaches have successfully been developed (cf. Bobrow, 1984). The component centered approach models the world as consisting of components that manipulate materials and conduits that communicate between components. The process centered approach uses views, describing static properties, and processes, describing the changes, to build models. Finally, the constraint centered approach does not commit itself to a specific modeling ontology, but has a fixed set of constraints that can be used to created qualitative differential equations that represent the behaviour of a real-world system.

The only well known existing ITS that uses a qualitative model for its simulation is SOPHIE III (Brown et al. 1982), a reactive learning environment for teaching troubleshooting in electronic circuits, of which the coaching module was never realized.[1] More recently White and Frederiksen (1987) describe their work on QUEST, an instructional system for the same task, based on a progression of mental models, from a qualitative to a quantitative one. Both systems are based upon a specific subset of the component oriented approach.

However, to improve the use of qualitative reasoning techniques for teaching purposes the current state of the art needs to be extended in two ways. First, the theory underlying the three approaches must be unified into a single conceptual framework (cf. Bredeweg and Wielinga, 1988; Bredeweg, 1989), which makes it possible to represent and use different *types* of models of the physical world in a single ITS. Secondly, this conceptual framework must be represented explicitly in order to facilitate access to the entities from the simulation at a level that closely matches the human interpretation of the simulation. We present an integrated qualitative reasoning approach, implemented as a shell called GARP that can be used for both modelling of the domain (the simulation) and the student, not only to reflect the learning (cf. QUEST where the different models are meant to specifically do that) but also for diagnosing lacks of knowledge or alternative conceptions. This approach will be discussed in the next sections.

Types of Knowledge in Qualitative Reasoning

The notion of different types of knowledge is based on the premiss that knowledge used by people during problem solving can be distinguished according to the different roles the knowledge plays in the reasoning process. Similar to Breuker and Wielinga (1989) we identify four types of knowledge. The first type of knowledge describes the domain knowledge of a particular domain. The knowledge of this type is static in nature, i.e., it is not specified what control regime is needed in order to use the knowledge effectively. The second type of knowledge concerns the basic problem solving actions out of which a reasoning process is built. Problem solving actions are elementary in the sense that other parts of the problem solver can not influence their internal control. Moreover the problem solving actions impose a specific use on the domain knowledge, which we will refer to as the *role* the domain knowledge plays in the inference process. The third type of knowledge describes what goals are involved in a particular problem solving task, and what the possible ways of satisfying a goal are (that is what sequence of problem solving actions is needed to reach a certain goal). Finally, the fourth type of knowledge concerns strategic knowledge. It describes the overall control of the reasoning process.

To model a students knowledge about a specific system from the real-world the domain knowledge, together with how this knowledge can be used in the reasoning process, is of crucial importance. In this paper we will not address issues concerning task and strategic knowledge (but see for instance: Winkels et al. 1989).

Inference Actions and Use of Domain Knowledge

The static domain knowledge for qualitative reasoning encompasses descriptions of physical objects, properties of these objects, processes, component models, constraints between properties, etc. The original approaches to qualitative reasoning provide specific ontologies about how the knowledge at the domain layer can be structured. The inference layer abstracts from the domain specific knowledge by 1) describing the *canonical* inferences used in the reasoning process and 2) pointing out the *role* the domain knowledge plays in this reasoning process. The roles are described in Table 1.

Table 1

Roles of the Domain Knowledge in Qualitative Reasoning

Role name	Description of the role
System model description	A period of time in which the qualitative behaviour of the system does not change (SMD)
System elements	Entities (like physical objects) from the real-world explicitly represented in the problem solving process
Parameters	The properties of system elements
Parameter values	The values a parameter can have
Quantity spaces	An ordered set of values that a specific parameter can have
Parameter relations	Dependencies between parameters
Qualitative calculi	The semantics of a parameter relation
Mathematical model	A set of relations that holds during a particular SMD
Partial models	Partial models that are used to further specify SMD's
Transformation rules	Knowledge about how to find successive states of behaviour
Behaviour descriptions	A set of SMD's ordered in time. It represents the possible behaviour of some real-world system

Given this list of roles, the canonical inferences used in qualitative reasoning can be described. Figure 1 gives an overview of these inferences.

Starting with the initial description of a system from the real-world (= input system) the purpose of the *compound specify* inference is to derive all the behaviour properties of that system. Two inferences must be carried out in order to arrive at such a description, *assembling* all the partial models that apply to the input system, and *computing* the mathematical model. Computation implies both deriving new values and checking the consistency of the parameter relations. The *compound transformation* inference is concerned with identifying successive states of behaviour. Termination rules specify the conditions under which a particular state of behaviour will terminate and are used to *select* the possible terminations (for example: the temperature was increasing and has now reached the boiling point). Precedence rules specify the order in which changes take place and are used to *sort* the available terminations. Finally, transition rules specify the conditions that need to be satisfied by the new state of behaviour in order to be a valid successor (for example: parameter values that were not selected by the termination rules may not change their values in the next SMD). They are, together with the ordered terminations, the input for *transforming* the current SMD into its successor.

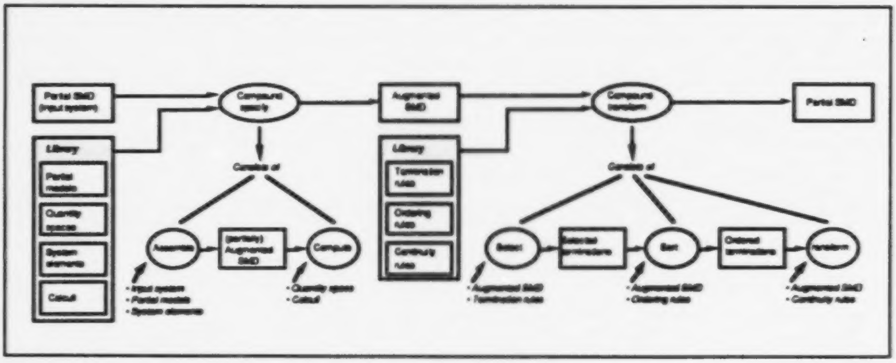
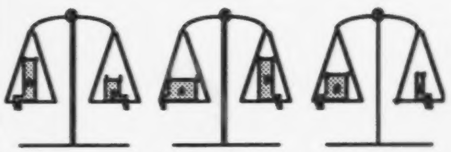

Partial SMD (input system)
Compound specify
Augmented SMD
Compound transform
Partial SMD
Library
Partial models
Quantity spaces
System elements
Calculi
Consists of
Assemble
(partially) Augmented SMD
Compute
- Input system
- Partial models
- System elements
- Quantity spaces
- Calculi
Library
Termination rules
Ordering rules
Continuity rules
Consists of
Select
Selected terminations
Sort
Ordered terminations
Transform
- Augmented SMD
- Termination rules
- Augmented SMD
- Ordering rules
- Augmented SMD
- Continuity rules

Figure 1. The inference structure for qualitative reasoning

Examples of Student Modelling

To see whether the modelling language of GARP complies with the way people reason about physical systems, and whether it can be used to model alternative conceptions of students, we had several students think-aloud while solving some prediction problems concerning balances. Some examples are depicted in Figure 2.



Balances with containers filled with water on each arm. Because of outlets near the bottom of the containers the water gradually flows out. The 2 containers differ in shape and in the amount of water they contain.

Figure 2. Examples of the balance problems

Comparing the problem solving activities as manifested by human subjects in the think-aloud protocols with those predicted by GARP, supported the "cognitive plausibility" hypothesis of the conceptual framework on which GARP is based (Bredeweg and Schut, 1991).

BEST COPY AVAILABLE

One of the models often used by students in the initial phase of the experiment was, what we refer to as, a *context independent* model of the balance problem. Being faced with an unfamiliar problem students apparently start with a detailed analyses. Each physical entity (= system element) of the problem is considered carefully and studied independent from its context. In Table 2 below some of the important entities of the domain knowledge of such a detailed model are given.

Table 2
The Context-Independent Balance Model

system elements	param-eters	par. values	par. relations	partial models
container	pressure	zero	press -> flow-rate	liquid-flow
liquid	amount	minus	height -> pressure	container-view
outlet	mass	plus	amount = mass	contained-liquid
balance-arm (L R)	volume	low	amount = volume	balance-view
balance	flow-rate	high		liquid
contain-relation	level			
support-relation	height			
connect-relation	width			
	position			

After solving some of the balance problems students gradually acquired a less detailed model which they could effectively use to predict the balance behaviour. They abstracted from the detailed physical structure by viewing each balance-arm as a "thing" *having* a certain mass and *losing* a certain mass. In fact they created, what we refer to as, a *context dependent* model by removing all the details that were irrelevant for the specific context in which the balance configuration operated. Table 3 below shows some of the important entities of the domain knowledge of this abstracted model.

Table 3
The Context-Dependent Balance Model

system elements	param-eters	par. values	par. relations	partial models
mass-looser (L R)	mass	zero	height -> flow-rate	balance-view
balance	mass-loss	minus		mass-looser-view
	height	plus		
	position	low		
	flow-rate	high		

In the context dependent as well as in the context independent model different viewpoints can exist (including misconceptions) with respect to the factors that students think determine the speed of the liquid outflow (or mass-loss): 1) size of the water-outlet, 2) height of the water-column, 3) total amount of water (its volume), and 4) width of the water-column (its surface). Each viewpoint can easily be represented by relating the influence on the flow-rate to the "appropriate" parameters in a partial model that describes the interaction between the container and the liquid it contains. GARP can reason with these alternative viewpoints and as such provide *explanations* for student responses (i.e., be used for diagnosis) or *predict* them, which means it can be used to select critical problems for diagnostic or teaching purposes.[2]

Concluding Remarks

In this paper we have presented a qualitative reasoning shell, GARP, that represents and simulates the (physical) world at the conceptual level (instead of at the mathematical level) and which is therefore better suited to be used within an ITS, both for modeling the domain and for representing alternative conceptions of students. This is so because of the explicit representation in the qualitative reasoning shell of the conceptual framework on which GARP is based, and the observation that it maps rather well on the representations humans seem to use. GARP can reason with different *types* of qualitative models (component, process, or constraint centered) and with different conceptions of a particular system (including wrong ones). This makes the domain model very suitable for generating different kinds of explanations and for doing diagnosis. Finally, we can use the Student Models and the different Domain models for generating critical tests to induce learning or awareness of misconceptions on the part of the student (cf. Socratic Dialogue, Stevens and Collins, 1977).

Notes

1. Note that though SOPHIE III is based on a qualitative approach, it does not have true causal models of the circuits, but uses circuit specific rules and links.
2. White and Frederiksen (1987) suggest selecting a problem the student cannot solve with her present model, but which can be solved using the "next" model in QUEST to invoke learning.

Acknowledgement: We would like to acknowledge the contribution of Cis Schut to the research presented here.

References

Bobrow, D.G., (Ed.). (1984). *Qualitative reasoning about physical dystems*. Amsterdam: Elseviers Science Publishers B.V.

Bredeweg, B. (1989). Introducing meta-levels to qualitative reasoning. *Applied Artificial Intelligence* 3(2):85-100.

Bredeweg, B. and Wielinga, B.J. (1988). Integrating qualitative reasoning approaches. In: *Proceedings of ECAI-88*, pp. 195-201, Munich.

Bredeweg, B. and Schut, C. (1991). Cognitive plausibility of a conceptual framework (for modeling problem solving expertise). Technical report, University of Amsterdam.

Breuker, J.A. and Wielinga, B.J. (1989). Model driven knowledge acquisition. In: P. Guida and G. Tasso (Eds), *Topics in the design of expert systems*, pp. 265-296. Amsterdam: North Holland.

Brown, J.S., Burton, R.R. and deKleer, J. (1982). Pedagogical, natural language and knowledge engineering techniques in SOPHIE I, II and III. In: D. Sleeman and J.S. Brown (Eds), *Intelligent tutoring systems*. New York: Academic Press.

Bruner, J.S. (1961). The act of discovery. *Harvard Educational Review* 31:21-31.

de Jong, T., (Ed.). (1991). Computer simulations in an instructional context. To appear in *Education and Computing* 6 (special issue). Amsterdam: Elsevier Science Publishers.

Elsom-Cook, (Ed.). (1990). *Guided discovery tutoring*. London: Paul Chapman Publishing Ltd.

Hollan, J.D., Hutchins, E.L. and Weizenbaum, L. (1984). STEAMER: an interactive inspectable simulation-based training system. *AI Magazine* 5:15-27, 2.

Lavoie, D.R. and Good, R. (1988). The nature and use of prediction skills in a biological computer simulation. *Journal of Research in Science Teaching* 25:335-360, 5.

Stevens, A.L. and Collins, A. (1977). The goal structure of a Socratic Tutor. In *Proceedings of ACM 1977*, pp. 256-263.

White, B.Y. and Frederiksen, J.R. (1987). Qualitative models and intelligent learning environments. In R.W. Lawler and M. Yazdani (Eds.), *Artificial intelligence and education, volume 1*, pp. 281-305. Norwood: Ablex Publishing.

Winkels, R.G.F., Achthoven, W.F. and van Gennip, A. (1989). Methodology and modularity in ITS design. In *Proceedings of AI & ED-89*, pp. 314-322.

Woolf, B. and Cunningham, P.A. (1987). Multiple knowledge sources in intelligent tutoring systems. In *IEEE Expert*, pp. 41-54.

Learning Construction Problem Solving

Joost Breuker
Dept. of Social Science Informatics, University of Amsterdam
Roeterstraat 15, 1018 WB Amsterdam, The Netherlands

Abstract: A major thesis of this paper is that learning problem solving in construction tasks is different from that in analytic tasks, because they do not dependent upon the same types of knowledge that controls the problem solving. The methods in construction tasks–e.g., design–are more domain dependent than in analytic tasks, as diagnosis. Moreover, in analytic tasks application knowledge can be derived from 'first principles.' Therefore, skills can be acquired effectively in an 'understanding-practice' paradigm. However, in design tasks the methods only partially reflect domain theories, and have a more heuristic, and incomplete character. They evolve from culturally accumulated experience. Learning to solve problems should be based upon exemplary cases, because there is no way to understand the construction principles in a comprehensive way. The parallel between domain evolvement and skill development can be expressed in genetic terms that ontogenetic development follows phylogentic development, i.e., evolution of a species.

Nothing is as Practical as a Good Theory...

Traditionally, insights and cognitive skills have been viewed as the two major components of human cognition. Insight is acquired by 'being told,' and by exploration, while the appropriate way to become skilled is by 'doing,' like in the acquisition of motor skills. Insight comprises the knowledge of the structure and working of a system, i.e., 'what-is-in-a-thing' and how things work. I will use the term system as almost synonymous with domain, assuming that any domain in fact refers to some system in the world (Clancey,1985). This system may be artifactual (device), imaginary (mathematics), or natural (e.g., the human body). Cognitive skill is knowledge how to apply domain knowledge to problems. This may be based on explicit reasoning about a situation, or it may consist of pattern recognition like inferences, i.e., expertise.

In traditional teaching the sequence of acquiring insight, stepwise application of this insight, and refinement by experience is followed to become an expert. That works well with analytic tasks–e.g., diagnosis–but I will argue that another paradigm should be used for construction tasks–e.g., design (see Construction Problem Solving).

From Insight to Applying Knowledge

In AI terms, insight consists of a coherent domain theory, and an ontology which is the grammar for this theory. The ontology specifies how the elements of the theory can be interpreted, and is the basis of an 'inference engine' that can emulate all the behaviour (states) of the system at some level of abstraction, as e.g., in qualitative reasoning.

68

Understanding means then the identification of a history of actions which has transformed the system-in-the-world from some initial state into a given state. In principle, these components are sufficient to solve problems, as long as the theory about the domain is a good emulation of the system in the world. For instance, diagnosis is the tracing of an (anomalous) state, or element that explains the systems incorrect behaviour; design is the composition of a system that produces the required behaviour.

However, this view is hardly of practical use in problem solving. Except for trivial cases, solving any problem becomes intractable, as is well illustrated by playing chess on the basis of first principles, i.e., the rules. There are two ways out of this, both based on the classical principle of divide and conquer.

- The first one is to partition the model into specialised models. Steels (1990) proposes to view domain knowledge as consisting of underlying domain *theory*, which is a full description of a system based on 'first principles' (see for an elaboration in the KADS methodology, Wielinga et al., to appear). The second component consists of domain *models*, i.e., specialised abstractions from or views on the domain theory. These domain models contain typed knowledge, for instance component, taxonomic, functional, topological etc. structures, and limit the problem space.

- The second way to limit this space is by a *task decomposition* which specifies which successive intermediary states (goals) should be obtained, thus providing more local control to the inferences.

In effective problem solving both are combined, as they should as well in an effective teaching strategy. For instance, in teaching diagnosis in physiotherapy, the PHYSIODISC ITS considers the domain to consist of models of anatomy—a global and a detailed one—of tissue types (nerves, bones, muscles, etc.), and of affections (lesions, infections, etc.), which drive and constrain the diagnosis in a way that allows the student at each moment to explore a manageable hypothesis space (Breuker & Winkels, 1991; Winkels et al., 1989). The global localisation (anatomy) constrains which types of tissues are affected, and the affections are also to a large extent tissue dependent.

The correspondence between facilitating human and machine problem solving is not based upon the same factors, but they have similar effects. For the computer the partitioning is useful to prevent combinatorial explosion. For the novice, exploring the problem (hypothesis) space may be less of a problem, because humans may use underlying associative or 'massively parallel' mechanisms to guide the search (Breuker, 1981). The novice's problem is to keep track of all hypotheses: they do not fit short term memory, nor are they well organised. The subtle difference between man and machine is problems in information management vs search resources. A hierarchical structure eases both types of problems.

Becoming Expert

Acquiring the competence to solve problems via insight in the domain is the traditional way of teaching—and probably the most powerful and generic one. However, it is not the only way, neither is it sufficient to acquire expertise. A novice who solves problems on the basis of insight may perform as well as an expert in terms of correct solutions (Kahney, 1984), but his speed may be many orders of magnitude slower. As a consequence of experience, he learns:

- Knowledge typing by discrimination.
- To compile out inference steps, i.e., make shortcuts on the basis of generalising over these steps (cf. explanation based generalisation).
- Domain refinements, idiosyncrasies, and local inconsistencies: domain theories are idealisations.
- Empirical associations, i.e., subjective likelihoods of coocurrence, associational shortcuts, patterns of data, etc.

All this learning by doing can be guided by coaching so that experiences may have accumulative effects. In formal education insight followed by practice is the standard approach. It is efficient, and allows for flexible problem solving competence, because the student may fall back on reasoning from first principles if the 'skilled' approach fails. Moreover, he may learn from his failures because he can reflect on these (Jansweijer et al., 1986). If however doing comes before understanding, the student may learn the trick, but the scope is limited. Even foreign language learning is highly facilitated by understanding explicitly the grammar (in students over 12 years) (Krashen, 1982).

Construction Problem Solving

A major typing of problem solving tasks is between analytic and synthetic (construction) tasks (Clancey, 1985). Analytic tasks are characterised by the fact that the solution for a problem is part of the domain knowledge. In diagnosing diseases, the diseases are known, and the task consists of identifying which of these is applicable to a patient. In troubleshooting the components of the device and their (correct) functioning are known. In practice these identification or classification tasks are the most common ones, but there are more than these; e.g., assessment, prediction etc. (see Breuker & Wielinga, 1989, for an extensive taxonomy). What these analytic tasks have in common is that they can in principle be solved by *understanding what has happened, or should have happened*, and that understanding still plays a major role, even in skilled expertise. For instance Boshuizen et al. (1988) show that in medical diagnosis the role of first principle knowledge even increases with the experience of general practitioners.

However in synthesis tasks understanding as tracing of what is the case with a system only plays a minor, evaluative role, because there is no system yet. It has to be constructed on the basis of specifications of its prospective behaviour. The solutions are not known, only the potential elements or structures of the solution. Therefore, construction tasks are more complex than analytic ones, if only because the solutions consist of combinations instead of single elements.

If viewed in this way, construction tasks are almost impossible. A blind generator would first propose a combination, next its behaviour under the required circumstances (input states, or initial states) would be generated, and tested whether it fits the functional requirements or not. Although this evaluation by generating the behaviours under varying circumstances can already easily explode, it is still analytic and can be based upon understanding (Brown & Chandrasekaran, 1989; see Breuker & denHaan, 1991 for a surprising example in testing traffic regulations). The major effort is in the propose part of the task, i.e., to reduce the number of combinations to feasible, preferably optimal ones.

The reduction of combinations to possible ones can be derived from first principles, i.e., the domain theory (Winkels et al., this volume). In programming a construct like

```
WHILE i=INTEGER AND p=(FOR i:=2 STEP 1 UNTIL 100 DO
(WHILE j:=j+1) i:=i/j DO NOT PRINT[“OK”,i]) ELSE PRINT[“OK”,i];
```

violates a number of constraints in which the programming elements can be combined, even if it looks ‘semantically’ like a feasible construction to print all primes between 2 and 100.

However, it would be waste of time to learn first the possible combinations. There are more reasons to learn feasible ones, or rather those that applicable to a particular type of problem (goal). By practicing simple programming constructs, e.g., ‘sentinel-process-read-while’ that matches simple requirements (averaging over varying numbers of numbers) practical higher level elements are acquired (Johnson & Soloway, 1987). This way of abstracting possible, respectively functional higher level elements fits with the understanding-practicing paradigm, but has a limited scope. It assumes that there is no scaling problem, and functional abstractions can be simply acquired, one level after the other. However, in complex domains this is not the case. First, the functional abstractions may have a very limited scope. Second the distance between requirements and the realisation of the construct (design, plan) may be too large.

The difference between programming and software engineering illustrates the point that aggregation into larger size elements has a limited effect. Therefore, other principles are used. Between specifications and the software exist mapping (transformational) relations, because they are qualitatively different, in a similar way as the drawings and specifications of an architect, and the plans of the builder are expressed in different terms. The use of qualitatively different abstraction layers, as often expressed by the use of *modelling languages*, is a second way of keeping the set of feasible solutions manageable. Each layer represents an abstract intermediary construction. In this way the construction task can be decomposed into *stages*, consisting of transformational and/or refinement steps towards the construction of the solution. Stages and modelling languages form the backbone of design methodologies (Breuker & Wielinga, 1989).

The third way to constrain the set of solutions is orthogonal to the use of modelling languages. Working backward from successful constructions, their structure can be decomposed in higher level parts, each of which can be designed independently. Object decompositions and modelling languages can be combined. For instance, the user interface can be specified more or less independently from the data processing components. I will come back to this.

Dependent on the degree to which abstraction layers and standard decompositions are available, construction tasks become more and more routine, and ‘degrade’ towards analytic tasks. Brown & Chandrasekaran (1989) distinguish three ordinal classes of design, dependent on the available routine knowledge. However, not only the available knowledge, but also the task features can make construction task more or less complex, because parts, or features of the solution are already given. Kruger (1990) classifies construction tasks according to which elements and/or structures are ‘given’ in the input (see Figure 1).

I will not elaborate on these classifications, but the point here is that construction tasks are more heterogenous than analytic tasks. Despite these obvious differences it turns out that the global task structure for all these tasks is the same. Invariably empirical investigations show an analysis stage, in which the problem specification is classified, or transformed in order to allow decomposition of the problem, which is followed by an assembly (synthesis) of the solutions. This is essentially also what Descartes’ (1637) famous method prescribed. This pattern may be repeated per stage when intermediary constructions can be made. This standard approach, which hides so much heterogenity, suggests that

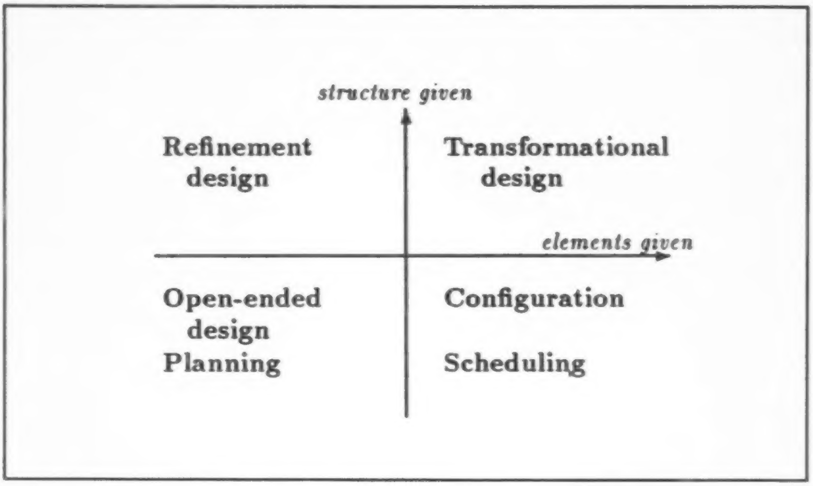


Figure 1. Classification of construction tasks according to what is given

the task structure is of no 'guiding' importance, and rather an epiphenomenon of applying domain dependent methods.

If that is the case, it appears of no use to teach a student explicitly the successive steps to arrive at solutions—which works quite well in analytic domains—because they are vacuous, but rather teach him the domain knowledge that allows him to plan their activities.

However, this domain knowledge is not derivable from the domain theory. Abstraction layers and decompositions—i.e., *design hierarchies*—have not been obtained in a principled, but in a practical, heuristic way (see below) and are 'incomplete', because it cannot be proven that they are sufficient for obtaining correct solutions. The decomposition may still lead to unforeseen interactions when the solution is synthesised; similarly, the abstraction layers may hide properties that may conflict at lower levels. In other words, the student cannot *understand what is happening* when partial or abstract solutions are assembled, and/or refined. He has to learn to live with heuristic notions, that cannot be completely acquired in a 'bottom-up' way via domain theories (understanding). The heuristic nature leaves a degree of art in the engineering. In return for lack of predictability, original and elegant solutions may emerge, so that new heuristics may grow. Therefore a different instruction paradigm seems appropriate that maintains this heuristic flavour. In this paradigm, between acquiring insight and practicing there are two heuristic knowledge sources: exemplary cases and methodology. Because this knowledge is to a large extent experiential, one could take the (best) experiences in the domain as the guideline for the curriculum. This heuristic in planning a curriculum can be paraphrased in genetic terms that the ontogenetic development follows the phylogenetic one.

Genetics of Construction Tasks

The abstraction and decomposition heuristics—'design hierarchies' (Brown & Chandrasekaran, 1989)—develop very slowly. Evolution of a species may serve as a metaphor for most of the stages.

1. *Invention*. A first adequate solution is found, c.f. a viable organism exists.

2. *Imitation*. The solutions are imitated and tested in variable circumstances, c.f. a reproducing organism.

3. *Experimentation*. Local variants are constructed to meet a broader range of circumstances, c.f. genetic mutations.

4. *Generalisation*. The scope of the variants is defined by assessing their relative effects over ranges of circumstances, c.f. genetic cross fertilisation. At this stage the domain has proven its practical applicability, c.f. within a niche the species can survive.

5. *Abstraction*. Thus far the domain theory has played only a local role, consisting mainly of locally applicable theories with too general a scope (e.g., mechanics-in-general), but with experiential, case based overlays. Abstraction may focus on the nature of these overlays, and find common domain features that seem to be relevant, which may be located to different abstraction layers, and for each layer a coherent theory (modelling language) may emerge or be enforced. Also, common structures in the object are identified that allow hierarchical decomposition. Here the metaphor becomes less genetic: a species forms societies with cooperation and specialisation.

6. *Reflection and formalisation*. By applying formalisms (mathematics) to layers and structures, they can be fitted in a common framework and relations can be 'calculated' in an algorithmic way. Construction is not only 'degraded' to analysis, but may also become largely automised, because the domain theory is sufficiently complete. Here is where the metaphor should stop.

Of course, all these stages overlap, and iterate, because new materials, techniques, etc. introduce 'new species of the same family.' In fact, the last stage may never be reached or completed: not only because it is easily beyond the state of the art in science, but also because of external evolutions, both in requirements and techniques.

A curriculum based on this view would insert between the acquisition of insight, and the application and practicing of the task, stages that mimic the major steps. The examples are from knowledge engineering.

Insight: Domain theories of the constituting disciplines are acquired: e.g., AI supported by computer science, cognitive psychology, etc.

A classical case: A both prototypical and elegant solution is explained, demonstrated and/or 'reconstructed'; the emphasis is on the solution, rather than how it is obtained—e.g., MYCIN, including NEOMYCIN, and other rational reconstructions (Cendrowska & Bramer, 1984)(c.f. imitation and experimentation).

Families of cases compared: The standard domain solutions are compared and classified (e.g., Internist, R1, Dipmeter Advisor, etc.)(c.f. generalisation).

Methods, techniques and methodologies: The how-to-construct-it is instructed. This may range from supportive techniques and methods to fully integrated methodologies—e.g., knowledge elicitation, data analysis (modelling languages), life cycle models, KADS methodology (Breuker & Wielinga, 1989), etc. (c.f. abstraction).

Projects and practice: Construction tasks are practiced and coached, for instance in the form of projects that increase in complexity (e.g., developing small knowledge based systems in student projects).

This sequence of activities may not look very revolutionary, nor illuminating. It has the same vacuous flavour as the prescriptive how-to-construct-it task structures. However, in designing curricula for knowledge engineering (Wielinga & Breuker, 1986) and for constructing regulations (designing laws) this evolutionary heuristic has lead to at least a coherent view on how the ingredients should be selected and ordered.

References

Boshuizen, H.P.A., Schmidt, H.G. and Coughlin, L.D. (1988). On the application of basic-science knowledge in clinical reasoning: implications for structural differences in knowledge between experts and novices. In: *Proceedings of the 10th Annual Conference of the Cognitive Science Society*, Hillsdale, NJ, Erlbaum.

Breuker, J.A. and den Haan, N. (1991). Separating world and regulation knowledge: where is the logic? In M.Sergot, editor, *Proceedings of the 3rd Conference on AI and Law*, New York, ACM.

Breuker, J.A. and Wielinga, B.J. (1989). Model Driven Knowledge Acquisition. In P. Guida and G. Tasso, editors, *Topics in Expert Systems Design* 265-296, Amsterdam, North Holland.

Breuker, J.A. and Winkels, R. (1991). The use of the KADS methodology in designing an ITS for diagnosis in physiotherapy. In J. Talmon, editor, *System engineering in medicine*. Springer, Berlin.

Breuker, J.A. (1981). Availability of knowledge. Ph.D. Thesis JB-01-81, COWO University of Amsterdam.

Brown, D.C. and Chandrasekaran, B. (1989). *Design problem solving: knowledge structures and control strategies*. Pitman, London

Cendrowska, J. and Bramer, M. (1984). Inside an expert system: A rational reconstruction of the Mycin consultation system. In T. O'Shea and M. Eisenstadt, editors, *Artificial Intelligence*. Harper & Row, New York.

Clancey, W.J. (1985). Acquiring, representing and evaluating a competence model of diagnostic strategy. In Chi, Glaser, and Far, editors, *Contributions to the Nature of Expertise*

Descartes, R. (1637). *Discours de la Methode*.

Jansweijer, W., Elshout, J., and Wielinga, B. (1986). The expertise of novice problem solvers. In *Proceedings ECAI'86*. Brigthon.

Johnson, W.L. and Soloway, E. (1987). PROUST: an automatic debugger for Pascal programs. In G. Kearsley, editor, *Artificial intelligence and instruction* 47-68. Addison-Wesley, Reading, MA

Kahney, H. (1984). *Novice programmer's problem solving*. PhD thesis, Open University, GB.

Krashen, St.D. (1982). *Principles and practice in second language acquisition*. Pergamon Press, New York.

Kruger, C. (1990). Interpretatiemodellen voor synthesetaken. Master's thesis, University of Amsterdam, SWI, 1990.

Steels, L. (1990). Components of expertise. *AI Magazine*, (Summer):28-49, 1990.

Wielinga, B.J. and Breuker, J.A. (1986). Training knowledge engineers using a structured methodology. In Th. Bernold, editor *Expert systems and knowledge engineering* 133-140. North Holland, Amsterdam.

Wielinga, B.J., Schreiber, G. and Breuker, J.A. (to appear) KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*.

Winkels, R., Breuker, J.A. and den Haan, N. (this volume). Principles and practice of domain representation in EUROHELP.

Winkels, R.G.F., Achthoven, W. and van Gennip, A. Methodology and modularity in ITS design. In D. Bierman, J.A. Breuker, and J.A.C. Sandberg, editors, *Proceedings of the 4th International Conference on Artificial Intelligence and Education* 314-322, Amsterdam, IOS.

Entrenched Beliefs, Inconsistent Information, and Knowledge Change

William F. Brewer
Department of Psychology
University of Illinois at Urbana-Champaign
603 East Daniel St., Champaign, IL 61820

Clark A. Chinn
Center for the Study of Reading
University of Illinois at Urbana-Champaign
51 Gerty Drive, Champaign, IL 61820

Abstract: Entrenched beliefs are beliefs that are strongly bound in a web of observational and/or theoretical support. Previous research on children's development of naive models in the domain of astronomy has suggested that these models are developed as children are slowly forced to give up a succession of entrenched beliefs. This paper reports preliminary data from a study that investigates how adults deal with information that violates entrenched beliefs in the domains of relativity and quantum mechanics. There were two principal findings. First, it is necessary to distinguish between knowledge and belief. Subjects were often able to understand ideas that they refused to believe. Second, adults tended to deal with information that contradicted their beliefs in one of several ways: by rejecting the information, by reinterpreting the information, or by making partial changes in their knowledge and beliefs though accommodation and compartmentalization. We think these reactions to contradictory information are the initial stages in a learning process that would eventually lead to a reorganization of knowledge similar to that found in children learning about observational astronomy.

Recently the members of the Illinois Knowledge Acquisition Project have been studying the development of naive models of the physical world in the domain of observational astronomy. We selected this domain because the scientific models in this area are clear, much of the data to support the models is available to the child, and because it is a domain in which historically there have been a number of major conceptual revolutions. We have investigated children's knowledge about topics such as the shape of the earth (Vosniadou and Brewer, 1991b) and the day/night cycle (Vosniadou and Brewer, 1991a). Our basic methodology has been to carry out extensive structured interviews with children that focus on their beliefs about the shape of the earth, the movement of the solar objects, etc. We have asked three basic types of questions: (a) Factual questions designed to test the child's knowledge of relevant facts, e.g., "Does the moon always look the same?" (b) Explanation questions designed to lead the child to provide an overt explanation of some information, e.g., "Where does the sun go at night?" (c) Generative questions designed to require the child to use previously acquired knowledge to generate an answer to a new problem, e.g., "Can people live on a star?"

Astronomy Results

The data show that most of the children in these studies have developed an internally consistent model of the shape of the earth. The youngest children (4-5 yrs.) tend to have flat earth models. The older children (10-11 yrs.) tend to have the culturally accepted model of a spherical earth. The children of intermediate age tend to have "synthetic" models which incorporate both flat earth beliefs and round earth beliefs. A common synthetic model is the hollow earth model. Children with this model believe that the earth is spherical, but that it is hollow with people living on flat land *inside* the sphere. Another common synthetic model is the dual earth model. Children with this model believe that there are two earths, a flat one on which we live and a round one which is up in the sky.

Theoretical Account of Astronomy Results

Vosniadou and Brewer suggest that the synthetic models are the products of children's attempts to reconcile the culturally accepted view of a spherical earth with the initial model of a flat earth derived from their own observations. They argue that the change from flat earth models to the mature spherical model is slow because the flat earth model is supported by a set of "entrenched beliefs" which children find very hard to give up. Examples of entrenched beliefs are the belief that the ground is flat and the belief that unsupported things fall. Vosniadou and Brewer point out that one can derive the obtained synthetic models by assuming that children start out with initial models which are consistent with their entrenched beliefs and then slowly over time they are forced to give up these beliefs. For example, children who have developed a hollow sphere model have given up the entrenched belief that the earth needs to be supported but have retained the view that the ground is flat.

Entrenched Beliefs

The term "entrenched" is taken from the philosopher Nelson Goodman (1965), but we use it in a somewhat different fashion. An entrenched belief is a belief that is strongly bound in a web of observational and/or theoretical support. Entrenched beliefs should be hard to change, since to modify them one must change beliefs that have much observational support and that enter into a wide range of successful theoretical accounts of the world. Entrenched beliefs can be seen both in naive models of the world (e.g., heat is a substance) and in the sophisticated models of modern science (e.g., energy is conserved).

Knowledge Change in Adults

The present paper reports some preliminary results from a new research project investigating knowledge change in adults. The present study has two complementary goals: We want to introduce the construct of "entrenched" beliefs to the study of learning in adults, and we want to use the findings from adults to give a more detailed account of the global changes that we have seen in children.

Previous Laboratory Studies of Knowledge Change

In a paper on psychological studies of concept acquisition, Ahn and Brewer (in press) review the literature and argue that many previous studies are limited by the nature of the materials used. They point out that most of the studies in this area have used simplified experimental materials such as dot patterns or lists of individuals who are members of a club. Ahn and Brewer advocate the use of materials that are complex enough to allow the subjects to generate explanations for the presented information. We agree with the argument for the use of more complex materials, but want to extent it a step further to advocate the use of material with the potential to contradict the entrenched beliefs of the learner.

Domain Selection

In this study we wanted to find materials that would place adults in somewhat the same circumstances in which children find themselves as they try to understand what adults mean when they say that the earth is an enormous sphere rotating in space. We think that educated adults in developed societies have given up many entrenched beliefs and have accepted many of the counter-intuitive aspects of modern science. We examined a wide range of modern scientific theories and decided that two areas of modern physics, quantum mechanics and special relativity, still had the potential to violate a number of entrenched beliefs in adults. Physicists discussing quantum mechanics and relativity frequently point out the problems involved in understanding these topics. For example, in a discussion of quantum mechanics and relativity theory March (1978) notes, "Both theories contain startling concepts that seem absurd or paradoxical, for they conflict with basic intuitive feelings about space and time, cause and effect. (p. 3)." Pagels (1983) points out that, "The way in which physicists and others have trouble grasping quantum reality reminds me of the way children respond when confronting a concept they do not yet grasp. (p. 48)." Therefore, we predicted that texts describing relativity and quantum mechanics would violate the entrenched beliefs of undergraduate readers.

Method

Our basic method was to have undergraduate subjects read sections of text that provided nonmathematical treatments of quantum mechanics or special relativity. The text on relativity was approximately 2,400 words long and was based on March (1978) and Gardner (1976). The text on quantum mechanics was approximately 3,200 words long and was based on Peat (1990) and on Davies and Brown (1986). After reading the text each subject was asked a series of questions, some factual, some explanatory, and some generative. There were 28 subjects; half read the text on relativity and half read the text on quantum mechanics. There were two set of questions for each topic. Half of the subjects reading each passage received one set of questions and half received the other set of questions. Each subject also received a questionnaire asking about their background in physics. Preliminary results for the relativity texts are given below.

Results

Knowledge vs. Belief

Our initial pilot work revealed that materials of the type we used lead to a most interesting phenomenon. Our subjects frequently would try to give an answer derived from the theory described in the text but then state that they "sure didn't believe it." With the simplified material typically used in studies of knowledge acquisition in adults this distinction makes little sense. However, the distinction makes perfect sense for subjects attempting to master our texts. When our subjects use the word "belief" they apparently mean knowledge that is compatible with knowledge that they have stored in long-term memory. A half-hour of study is enough for our subjects to understand some aspects of the theory presented, but is not likely to be enough time to cause our subjects to give up their entrenched beliefs and integrate the new information with their previous knowledge, so they say they simply do not believe it. We think that statements of this type are one important sign that one is dealing with text that violates entrenched beliefs.

Entrenched Beliefs

We hypothesized that the relativity text would violate three entrenched beliefs. First, in the everyday world motions of objects are additive. If a boy riding in a convertible traveling at 60 mph throws a ball forward at 20 mph, the ball will travel at 80 mph relative to the ground. The text, however, explained that the ordinary principle of additive velocities breaks down at speeds approaching the speed of light, that light always has a constant velocity, and that nothing can travel faster than the speed of light.

Second, people are likely to have a deep-seated belief that properties such as length and the flow of time are constant. The text attempted to change the readers' beliefs on this issue with examples and demonstrations, showing that observers moving with respect to each other perceive each other to be thinner and to have watches that are running slow.

Finally, we postulated that people will hold that there is a privileged frame of reference. The relativity text argued that there are no privileged frames of reference. If a train is speeding past a stationary platform, it is not possible to say which frame of reference–the platform or the train–is "basic" or "correct." Indeed, it is not even possible to say that one frame is at rest and another is moving. One could just as easily say that the train is stationary and the platform is speeding backwards. We expected, however, that subjects would believe that "stationary" frames are privileged and that the stationary observer has the correct view of reality.

Evidence for Violation of Entrenched Beliefs

Students' responses to questions on relativity indicated that the relativity text did indeed violate their beliefs. For example, one question asked explicitly whether stationary frames of reference are the correct way to view the world. Many students gave responses such as: "Yes. If too many people look at too many perspectives, everything becomes relative to the perspective taken." And "Yes, because it will make things easier for everybody. There will be one frame of reference, and it will depend on the person's frame of reference." Most subjects were unwilling to concede that it makes no sense to talk about a correct frame of reference.

The Problem of Contradictory Information

Our texts were designed to provide information that contradicted the entrenched beliefs of individuals attempting to learn the material. This raises a fundamental question: How do learners deal with contradictory information? How do they react when they confront ideas that conflict with their own entrenched beliefs? Looking at the range of responses provided by subjects, we believe that there are several distinct ways in which a subject can respond to contradictory information.

Reject Information

One approach to contradictory information is to reject the material that contradicts entrenched beliefs. In this case the learner reads and processes the new information, but fails to incorporate the new information into any long-term knowledge or belief structures. A clear example of this approach is provided by a subject answering relativity problems dealing with the speed of light and the relative speed of objects at high velocities. Although much of the text was devoted to the idea that velocities are not additive at velocities approaching the speed of light, she answered questions as if velocities were additive. One problem explained that a spaceship was moving at 90% of the speed of light and fired a missile at 80% of the speed of light, relative to the spaceship. The question asked the speed at which the missile strikes its target. The subject's answer—both when she was answering from the relativity perspective and from her own perspective—was 170% of the speed of light. There was no indication that the information in the text had affected her knowledge or beliefs in the slightest.

Reinterpret Information

A second way to deal with contradictory information is to reinterpret the contradictory information so as to make it consistent with prior knowledge and beliefs. Instead of the new information having an impact on the subjects entrenched beliefs; it is itself modified by existing knowledge and beliefs. Several subjects appeared to do this in the relativity domain. One problem that followed the relativity text presented a paradox involving a car traveling through a garage with a door at each end. At rest, the car is slightly longer than the garage. The car then speeds through the garage at 50% of the speed of light. Subjects are asked if an attendant standing inside the garage will ever perceive that the car is entirely contained within the garage (the correct answer is yes) and whether the driver of the car will ever perceive that the car is entirely contained within the garage (the correct answer is no). The correct response requires consideration of how fast light travels from the car and garage doors to the observer's eyes. The text had presented several parallel examples, but in answering the question, several subjects appeared to reinterpret the relativistic examples as a human perception problem. One subject wrote, "He [the driver] is moving so fast that when he looks forward the front door is closed and if he immediately looks back he will see the back door has also closed." This subject reinterprets relativistic phenomena as simply due to observers' inability to turn their heads fast enough to see everything at once. He has reinterpreted the relativistic phenomena without having to modify his own beliefs.

Partial Change of Belief

A third way in which subjects can deal with inconsistent information is to make partial changes in existing knowledge or beliefs. In our data, there are two principal approaches to making partial changes: accommodation and compartmentalization.

Accommodation. We define accommodation as the attempt to incorporate new information into existing knowledge while preserving the core beliefs in existing knowledge. The relativity texts impelled most subjects to take alternative frames of reference into account as they answered questions. But at the same time, many of these subjects did not want to give up the idea that stationary frames of reference are somehow primary. As a result, some subjects argued that relativistic effects are illusions, with the at-rest frame of reference revealing the correct state of affairs. For example, one subject wrote that the car fitting into the garage "is merely an optical illusion." Ironically, the physics textbook in which we originally found the problem (March, 1978) presented the car-garage paradox to illustrate that relativity is *not* merely an optical illusion. But if subjects can maintain that relativistic phenomena arise from optical illusions, they can retain the core belief that stationary frames give the correct perceptions.

Compartmentalization. A clear example comes from subjects answering the questions about the spaceship shooting a missile. In their answers to generative questions about the speed of light viewed from different reference frames, several students consistently applied the principle that the velocity of light is not additive. But the same subjects also believed that the velocities of objects traveling at very high speeds were additive. Thus, these subjects appeared to accept that velocities are non-additive in the domain of light, but they refused to allow nonadditive velocities to impinge on the more mundane world of tangible objects.

Restructuring

The final way in which a person can respond to information that contradicts an entrenched belief is to accept the new information completely, thereby restructuring their knowledge. We think it is unlikely that the learning experience of reading a text for 30 min can accomplish fundamental knowledge restructuring; however, a clear answer to this issue awaits more extensive analysis and additional data.

Conclusions

We have found that texts that violate the reader's entrenched beliefs provide interesting insights into the interplay of old knowledge and new knowledge. The responses to questions on these texts demonstrate that some subjects either reject or reinterpret the new information and so show little learning. Other subjects show accommodation or compartmentalization of the new material, which indicates some change in their knowledge. Few, if any, of these subjects showed clear signs of giving up the entrenched beliefs that would be required to accept these new theories. We assume that with more instruction most of these students would eventually be forced to give up their entrenched beliefs and would show a fundamental change in their knowledge similar to that shown in the work of Vosniadou and Brewer (1991a; 1991b) on children's understanding of observational astronomy.

In our future research we plan to contrast the learning of texts that violate entrenched beliefs with texts that do not. We also intend to contrast approaches adopted by children and by adults in dealing with inconsistent information.

Acknowledgements: We would like to thank Ellen Brewer for comments on an earlier draft of this manuscript. This research was supported in part by the Office of Educational Research and Instruction under Cooperative Agreement OEG-0087-C1001 for the Reading Research and Education Center at the Center for the Study of Reading.

References

Ahn, W-K., and Brewer, W. F. (in press). Psychological studies of explanation-based learning. In G. DeJong (Ed.), *Studies in explanation-based learning*. Kluwer Academic Publishers.

Davies, P. C. W. and Brown, J. R. (Eds.). (1986). *The ghost in the atom: A discussion of the mysteries of quantum physics*. Cambridge: Cambridge University Press.

Gardner, M. (1976). *The relativity explosion*. NY: Vintage Books.

Goodman, N. (1965). *Fact, fiction, and forecast* (2nd ed.). Indianapolis, IN: Bobbs-Merrill.

March, R. H. (1978). *Physics for poets*. Chicago: Contemporary Books.

Pagels, H. R. (1983). *The cosmic code: Quantum physics as the language of nature*. NY: Bantam Books.

Peat, F. D. (1990). *Einstein's moon: Bell's theorem and the curious quest for quantum reality*. Chicago: Contemporary Books.

Vosniadou, S. and Brewer, W. F. (1991a). *Children's explanations of the alternation of day and night*. Manuscript in preparation.

Vosniadou, S. and Brewer, W. F. (1991b). *Mental models of the earth: A study of conceptual change in childhood*. Manuscript submitted for publication.

Cognitive Science, Artificial Intelligence, New Technologies: How to Cooperate for a Computer-Assisted Learning to Read System

Jérôme Bruneau[1], Annie Chambreuil[2], Michel Chambreuil[2], Thierry Chanier[1], Marcelle Dulin[2], Paul Lotin[1], and Patrick Nehemie[2]
[1]*Département de Linguistique, Université de Clermont-II, 34 avenue Carnot, 63000 Clermont-Ferrand, France*
[2]*Laboratoire d'Informatique, Université Clermont-II, 63177 Aubière, France*

Abstract: Through the presentation of an initial version of a computer-aided system for the learning of reading, SIAAL, we show how new perspectives on this task are opened up by considering it collaboratively from the point of view of aspects of cognitive science, AI and new information technologies. For a number of reasons, the stage of learning chosen corresponds to the very early experiences in the classroom. SIAAL1 plays the role of a multi-expert coach who helps the student with the learning task by managing the multiple knowledge resources in a collaborative way. These knowledge resources are viewed as projections of those that could be used by the teacher. Representations of pedagogical situations are built from every interaction with the learner. They are used to update the learner model, and, with different psycho-pedagogic factors taken into account, for guiding the conduct of a session.

Introduction

The SIAAL Project in Context

The learning to read and reading mastering activities encompass multiple situations (elimination of illiteracy, functional illiteracy, early learning stages,...). These numerous situations and their socioeconomic incidences, such as problems of information accessibility, school failures, have made learning to read a fundamental problem in all societies.

In the context of computer-based systems developed for research on learning to read, the SIAAL project[1] which we present here comes within the current general perspective of Intelligent Computer-Assisted Learning Environments (Chambreuil, 1990). It attempts to create a new medium for the support of learning by combining aspects of the disciplines of cognitive science, artificial intelligence, new information processing technologies. The reading situation chosen by the SIAAL project is characterized by three aspects:

(1) *initial stages of learning to read.* In the first instance the SIAAL1 system focuses particularly on the very beginning of the learning process (first grade beginners/cours préparatoire (CP) in French primary schools).

We have chosen this very early stage of learning for several reasons:
- it is a key period for schooling,
- it is a period for which we had at our disposal solid practical experience in addition to theoretical studies,
- for the learner it is the period of systematized discovery of a new knowledge domain, and of mobilization of newly assimilated multiple cognitive resources,
- moreover, this stage of learning to read allows to approach the task without the necessity of facing the very complex problem of natural language processing.

(2) *learning in a mediating environment*. In this situation, the proposed system appears purely as an additional tool available to the teacher and not as a surrogate teacher. This choice allows–through an analysis of the learner's reading activity–to distinguish between problems relating to natural language understanding and those relating to the reading activity itself.

(3) *applicability for children in normal schooling*. This means, in particular, that we do not have to make hypotheses–linked to reported deficiencies–on access to information not related to reading.

In the rest of this paper, except where specifically mentioned, we refer to this particular situation.

Learning to Read: A Cognitive Approach

In our approach to reading (and in the SIAAL project), we assume that the young child has a basic mastery of his/her native language. Without going into a precise description of current models of reading (Rumelhart and McClelland, 1981; Perfetti and Rieben, 1989; Seidenberg and McClelland, 1989), we emphasize the following basic points:

- reading is a complex activity,
- reading can be seen as a dual process: building of meaning and/or of acoustic chains from a visual object (written material),
- this activity relies on the interaction between text-based data and reader-based resources,
- the resources called upon by the learner are numerous: perceptual, linguistic, metalinguistic, cognitive, metacognitive, affective. They are not brought into play at the same time, but are activated specifically with different priorities depending on the reading goal, the difficulty of the text and the reader's competence and aptitudes.

In his/her learning process, the child will have to organize the different knowledge spaces and master their interrelations. He will also have to control how the resources collaborate to achieve a task.

We now develop the main aspects of reading and learning to read processes, referring, on the one hand to a number of current studies (Ehri, 1989; Ferreiro and Gomez-Palacio, 1982; Sprenger-Charolles, 1988; Rieben and Saada-Robert, 1989), and on the other hand to our own observations made during the period since 1984 in CP. We illustrate these aspects by referring to one of the main problems of the initial stage, that of word recognition.

The written language has its own structures. The child has to understand the rules and working of the written system. Thus, he/she has to see the word as a specific graphic object, and not as a drawing, for example. He/she also has to see the word as a complex object, made up of letters varying in sort, number and position.[2] The word itself is an element of more complex structures (phrases, sentences, texts). The child will have to understand the position and function of a word in these structures.

The written system corresponds to the spoken one. The child has to establish this relationship and learn the correspondences between the two systems. He/she has to identify recurring regularities within words: auditory units (syllables), visual units (letter clusters specific to a given language). He/she can read a word using visual or phonetic clues, or both.

The written language is related to meaning. The child will have to establish the relation between a word and its meaning(s).

The written language is, in itself, an object of knowledge. Through the stages of learning to read, the child builds metalinguistic knowledge about writing. His/her aptitude to identify spaces between words is developed simultaneously with his/her mastery of word concept.

All these elements of knowledge interact simultaneously in the construction of the child's knowledge. There is a particular developmental order. Acquisition models (Frith, 1985; Seymour and McGregor, 1984; Ehri, 1985; Gough and Juel, 1989) refer to different stages: logographical, phonological, orthographical. However, we prefer to speak—as Rieben (1989) does—of "prevailing strategies": through our own observations of first grade beginners, we have noted important inter- and intra-individual variability in strategies, acquisition rhythms, as well as in the balance of the different components between them.

One remarks that the construction of knowledge does not follow a linear progression of visible performances. A child who is able to identify a word through the "whole word" recognition may have difficulty when reading this word at a later stage. This is not necessarily due to a regression but can be the result of the acquisition of knowledge about word analysis and its relations to the spoken word.

General Conduct of the First Working Session

A working session with SIAAL1 (current first version of the system) is characterized by general pedagogical goals and by the types of activities offered to the learner.

The Pedagogical Goals

There are different types of pedagogical goals: for instance, goals giving rise to an evaluation at the very time of the session (immediate goals), goals giving rise to a familiarization session with evaluation being carried out in following sessions (postponed goals).

The first working session with SIAAL1 holds the following pedagogical goals as essential:

- matching written chains with phonic chains,
- identifying and recognizing some fifty words,
- mobilizing multiple strategies in this identification,
- doing a first analysis of the written chain and of the word, allowing us to put forward the regularities specific to the written language.

The Types of Activities of One Session

In the first working session, all the activities are based on the reading of a story. Seven types of activities are offered to the learner. Each activity type is associated with an interface window that has specific functionalities and types of objects which can be brought into play. In the following part, such a window will be called a "didactic workshop."

In the first didactic workshop (A1), the system provides the learner with a part of a story (a sequence defined by its discursive functionality). Each sentence of the sequence is simultaneously read to the student, displayed and illustrated by a drawing. By clicking on the displayed sentence, the learner can also get another reading of it. During the reading of a sentence, some phrases may be specifically marked (for instance, by highlighting).

The characteristics of this workshop interface are:[3]

- its constituents (sound, text, drawing),
- the spatial setting of those objects (in different parts of the window),
- the temporal setting of the objects (succession, simultaneity, overlapping),
- the structural relationships between the objects (highlighting during the reading, ...),
- the functionalities of action of the learner (clicking on a phrase or a sentence to get it read aloud, ...).

The second activity (A2) consists of checking the learner's comprehension of the story. Here again a specific window is used where one can manipulate sound, text, picture.

Activities (A3), (A4), (A5) and (A6) deal with the identification of (written) words or of regularities on words in different types of contexts :

- (A3) words to identify in the sentences of the story,
- (A4) isolated words in a "sound/text/illustration" association context,
- (A5) isolated words in a "sound/text" association context,
- (A6) words to replace inside a sentence.

The seventh type of activity (A7) offered to the learner is akin to a micro-world type activity. It consists in giving the learner access to the functionalities for using and updating a "didactic dictionary." The didactic dictionary associated to SIAAL1 is in itself a complex system. We will not tackle it further in the context of this paper.

SIAAL1 as a Multi-Expert Coach

SIAAL1 has been conceived as a "multi-expert coach" for computer-assisted learning, organizing the multiple data resources in a cooperative way. These resources are seen as projections (or extensions linked to the specific possibilities of the computer) of the knowledge resources which could be used by a teacher.

In the following part, we simply indicate some directions for analysis and use on the base of the resources available within the SIAAL project.

Sound Resources

Sound resources are characterized by the following aspects :

- specific characteristics, in the sense that the words or the sentences of the story match specific sound data, which can be analyzed as significant constituents in a reading activity (syllables or vowels in a word)

- the matching of items from other data resources (written text or illustrations). These associations may be made either from the sound resource or from items of other resources.
- functionalities for the computer system as a learning partner, in the sense that the sound resources are called up by the system for the reading of the texts, instructions are given for different activities and comments are made to the learner. But these resources may also be activated to answer the solicitations of the learner who, for instance, asks for a word or a sentence.

In other respects, the sound resources are also constituents of didactic workshops. They contribute to the characterization of pedagogical situations. In doing so, they tell us more about their own functionality. Thus the simultaneous presentation to a learner of a written word, its oral reading, a drawing associated to it, can facilitate the creation of correspondences. However that representation may also create a "cognitive overload" and in fact hinder the creation of the desired correspondences.

Knowledge About the Story

The system must be able to select the items of the "world of the story" which are pertinent to the current working session.

Knowledge about the story generates problems to present to the learner. There are two aspects to this. It allows, on one hand, the definition of the relevant questions likely to be put to the learner within the scope of activities (A2), (A3), (A4) and (A5); and, on the other hand, the specification of various parameters of problem in the quest for an answer associated to a given question. For instance, when one word is to be identified among others, this knowledge will permit to select the group of words according to various structural criteria, one of which is their links with the sequences of the story already presented. In the same light, it will permit a definition of the texts for an activity (A6).

According to another view, this knowledge permits the analysis of the learner's answers with respect to their various relations of relevance to the expected answers and to the information conveyed by the text: answers explicitly shown in the text, answers that are deducible from the data contained in the text and answers corresponding to what the learner is expected to know about the world of the story. This knowledge acts as a filter of the learner's answers.

Without standing in the realms of general linguistic knowledge, the SIAAL1 system simply aims at dynamically building the only knowledge of the world linked to the story in order to use it in the ways mentioned above. For instance, the questions deal with particular relations between the characters of the story : these relations will be managed dynamically during the conduct of the story.

Pedagogical Knowledge

Other resources deal with knowledge, which we will describe as psycho-pedagogical. The development of a working session will be guided from these resources (and from the student model). They are of different types but somehow characterize an organized space that will be set up through the definition of an effective activity proposed to the learner.

Some of these resources will relate to general psycho-pedagogical knowledge, which structures the working session. For instance, it will organize the transitions between story reading, comprehension checking and different types of activities A3 to A6. This type of knowledge will also distinguish between activities relating to the application of "old" acquired knowledge, and activities leading to the building of new knowledge.

In a way, one can find some sort of very specific items of knowledge on the domain of learning to read at the other end of this knowledge space. They are used in establishing the elements actually introduced in a given activity. For instance, in the case of a word identification workshop, they will be the ones that stipulate whether the other words given as options are to be "close" words or "distant" words (here again with different proximity criteria).

Thus, we have a twofold organization of the knowledge space: a hierarchical organization in strata, and for each stratum of the hierarchy a specific organization (contrasting or complementary structures).

Student Model

This resource aims at capturing four aspects of a student cognitive model (beyond an history of the successes and failures) (Clancey, 1986; Dillenbourg and Self, 1990; Self, 1988a, 1988b):

- the level of acquisitions for the target words in the session (in terms of "approached word," "recognized word," "assimilated word," "triggered word");
- the influence of communication media in the didactic workshop (making it easier, disturbing);
- the way he/she builds his/her knowing-how-to-read (in terms of reading strategies);
- student's hypotheses on the domain (in terms used by Ferreiro (1982), for instance).

This information about the learner, resulting from an analysis of his/her activities, may also be compared to models of "learner types" or "evolution types," rather like stereotype models (Rich, 1989; Kobsa, 1990) corresponding to type (but dynamic) schemes constituted by a teacher through his experience.

This knowledge of the learner will influence the development of a session at the different organizational levels. It will not only contribute to the dynamic management of the types of activities offered but also, at the very time of an actual activity, offer a choice of elements of this activity: for instance the choice of words already known by the learner within a given didactic workshop.

Didactic Situations

The dynamic management of the cooperation between the multiple resources is made possible by the foundation of "didactic situations." A didactic situation is a form of "instant snapshot" of an interaction between system and learner. It memorizes different factors likely to be relevant to the student modelling and to the control of the working session within the interaction. For instance, in an interaction related to a word identification activity including sound and graphic resources, the created didactic situation will memorize elements such as: the position of the word in the development of the session, the psycho-pedagogical factors affecting the situation, the type of didactic media used in the situation, the expected answer, the answer given by the student, structural elements about the written domain included in the situation (for instance, the other suggested words ...).

These didactic situations are stored and constitute an history of the working session. They are also regularly analysed (not necessarily at the time of their creation) by the mechanisms that are dynamically updating the student model, the psycho-pedagogic parameters which organise the working session, and the information on the story cited above.

Didactic situations may also be analyzed at different "depth levels" either when first building the student cognitive model (relating to stereotypical models) or in a revision of that building according to new convergence indices[4] (Wenger, 1987).

Working Out the Resources in a Prototype System

A first prototype of a session with the SIAAL1 system has been developed in Lisp on Macintosh. One of the goals of the prototype was to provide the team with a research tool allowing the management of multiple data resources. Another goal was to begin the creation of various didactic workshop windows.

Because of time constraints on development, the system has been built around an inference engine which handles different rule-based representations of one aspect of the resources and their interactivity. These rules dynamically manage various bases of facts representing another aspect of the data resources. Finally a third aspect of some data resources (linked to their application in the didactic workshop, or to the expertise of the written language) is represented in terms of objects.

Conclusions: Perspectives and Difficulties

One of the immediate evolutions of the SIAAL project currently investigated—will deal with its architecture. Relying on an initial study made in 1988/89, we will develop a new system around an architecture that will use a control-blackboard (Gasser and Huhms, 1989; Engelmore and Morgan, 1988) which will be explicitly organized towards the management of multi-expert resources.

SIAAL1 was more concerned with items of interaction/cooperation of these resources, than with the investigation of one of them seen in isolation. Throughout that necessary investigation, we shall particularly concentrate on psycho-pedagogical factors as they play a central and organizing part in the dynamics of the system. The remaining difficulty is to point out those factors and their incidences, not at a highly general level but with the accuracy required by particular activity situations.

If one thinks that these investigations rely on analyzes coming from the cognitive sciences, one must not conclude that it is a one-way cooperation. Our own experience of working regularly with practising teachers, and of a pure theoretical reflexion shows us that the specification and realization of such a prototype, even in the domain of reading, are particularly useful for bringing out a knowledge acquisition methodology.

Lastly, we will make a final remark more generally concerning the expertise of a domain and its use in a computer-assisted learning environment. In the system presented here, the expertise on reading and on learning of reading is not confined to one specific module (or to two if the teacher module is in charge of the learning expertise). The expertise appears rather as being distributed among the resources which are used. In the same way, the learner's acquisitions, the building of his/her knowledge about the reading sentences, also deal with distributed acquisitions whose gathering will constitute the desired mastery.

We will conclude by stating that we are left with many difficulties to overcome before we can use such systems in real pedagogical situations. However, the present state of each of the domains concerned and the opportunities for their improvement lead us to believe that in a near future, especially for a problem as important as learning to read, new tools to help learning will be integrated into the various educational structures (whether traditional or distance learning).

Acknowledgements: We would like to thank: Michael Pengelly for his criticisms on the paper; Delphine Reniè for the English translation, and Rachel Panckhurst for the final corrections; the "Inspection Acadèmique" of Clermont-Fd from the regional educational board for its support of our collaboration with the primary school of Manzat and the secondment of one of its teacher for the knowledge acquisition task.

Notes

1. Systeme Informatique d'Aide á l'Apprentissage de la Lecture: Computer-Assisted Learning to Read System.
2. A child may recognize some conventional letters in isolation but may not be able to relate them to the composition of a word.
3. All the other workshops have been developed according to the same specification principles.
4. When a teacher can recall elements in a situation he did not take into consideration before and which at a given time become relevant.

References

Chambreuil A.and Chambreuil M. (1990). Advanced Information Technologies : Prospect for Learning to Read. *EURIT'90*, Herning, Denmark.

Clancey W.J. (1986). Qualitative Student Models. *Annual Review of Computer Science*, vol 1.

Dillenbourg P.and Self J. (1990). A Framework for Learner Modeling. *Report for the NAT*LAB project.* EEC's DELTA Program; and *AI report, Computing Department*, University of Lancaster,UK.

Ehri L.C. (1989). Apprendre á Lire et á Ecrire les mots. in L. Rieben & C. Perfetti : *L'Apprenti Lecteur*. Neuchâtel: Delachaux et Niestlè.

Engelmore R.S.and Morgan A.J. (1988). *Blackboard Systems*. New York: Addison-Wesley.

Ferreiro E., Gomez-Palacio M. and al. (1982). *Analysis de las Perturbaciones en el Proceso Escomar de Apredizaje de la Lectura y Ecritura.* Mexico: Direction General del Education Especial.

Frith U. (1985). Beneath the Surface of Developmental Dyslexia. in K. Patterson et al.: *Surface Dyslexia*. London: Erlbaum.

Gasser L.and Huhms M.N. (1989). *Distributed Artificial Intelligence*. London: Morgan Kaufmann.

Gough P.B.and Juel C. (1989). Les Premiers [illegible] de la Reconnaissance de Mots. in (Perfetti and Rieben, 1989).

Kobsa A. (1990). Modelling the User's Conceptual Knowledge in BPG-MS, a User Modelling Shell System. *Computational Intelligence*, vol 6.

Perfetti C.and Rieben L. (1989). *L'Apprenti Lecteur*. Neuchatel: Delachaux et Niestlè.

Rieben L.and Saada-Robert M. (1989). Evolution et Differences Individuelles dans les Strategies de Recherche de Mots chez les Debutants Lecteurs. *Internal report* No 6, Universitè de Geneve.

Rich E. (1989). Stereotypes and User Modelling. in A. Kobsa & W. Wahlster : *User Models in Dialog Systems*. Berlin: Springer Verlag.

Rumelhart D.E.and McClelland J.L. (1981). Interactive Processing through Spreading Activation. In A.M.Lesgold & C.Perfetti: *Interactive Processes in Reading*. Hillsdale, Erlbaum.

Seidenberg M.S., McClelland J .L. (1989). A Distributed, Developmental Model of Word Recognition and Naming. *Psychological Review*. in press.

Self J. (1988a). Student models: What use are they? In Ercoli P.and Lewis R: *Artificial Intelligence Tools in Education*. Amsterdam: North-Holland.

Self J. (1988b). The Use of Belief Systems for Student Modeling. *Congrès Europèen IA et formation*, Lille.

Seymour P.H.K.and McGregor C.J. (1984). Developmental Dyslexia: a Cognitive Experimental Analysis of Phonological, Morphemic and Visual Impairments. *Cognitive Neuropsychology* , vol 1.

Singer H., Ruddell R.B. (1985). *Theoretical Models and Processes of Reading*. Newark: IRA.

Sprenger-Charolles L. (1988). *L'Apprentissage de la Lecture et ses Difficultès*. PhD dissertation, Universitè Paris 5.

Vivet M. (1988). Reasoned Explanations need Reasoning on Reasoning and Reasoning on the Student. in Ercoli P.and Lewis R: *Artificial Intelligence Tools in Education*. Amsterdam: North-Holland.

Wenger E. (1987). *Artificial Intelligence and Tutoring Systems*. Los Altos: Morgan Kaufmann.

Tuning the Ideal Student Model: Towards Intelligent Editing of ITS Models

Bart Burns, Wayne D. Gray, and E. Robert Radlinski
NYNEX Science & Technology Center
White Plains, NY 10604

Grace, an ITS for COBOL, is an effective practice-style tutor (Gray et al., 1991) that has resulted from the successful transfer of tutoring technology from Anderson's laboratory at Carnegie Mellon University to NYNEX (Gray & Atwood, 1991). Ironically, because Grace is intended to be a commercial product available for use in a wide variety of circumstances rather than a research tool, we find ourselves confronted with a host of research issues that the CMU tutors do not have to face. Research involving two of these issues will be discussed in this presentation.

First, is the issue of the grain-size of the productions used by the *ideal student model* (ISM). The ISM (Anderson, Boyle, Corbett, & Lewis,1990) is a rule-based representation of domain knowledge at a grain-size at which a good student (as opposed to a domain expert) might possess. While this approach works exceedingly well, all current ISMs have been carefully handcrafted and fine-tuned over generations of students. We do not have this luxury of time for Grace, nor do we expect to have it for our newer tutors.

The second issue involves adjusting the weights used to determine when a particular rule (chunk of knowledge) has been learned. Our current weights are inherited from CMU where they were heuristically determined and fine-tuned on the CMU student population. However, our experience with several different, and more varied populations of students using the COBOL tutor, has convinced us that the same weights cannot fit all populations or all students within a given population. Some means must be found of quickly determining and *individually* adjusting the weights.

Our conference presentation will discuss:

(1) the techniques we have developed to collect and tabulate large amounts of protocol data,
(2) the criteria used to flag productions whose grain-size (according to the criteria selected) are too small or too large; and
(3) the criteria for flagging students whose learning profiles do not match the weights selected.

Additionally, we hope to include a short report on our attempts to validate and/or revise our criteria for flagging grain-size discrepancies, as well as on our attempts to adjust the weights for individual students in near real time (between lessons).

References

Anderson, J. R., Boyle, C. F., Corbett, A. T., & Lewis, M. W. (1990). Cognitive modeling and intelligen t tutoring. *Artificial Intelligence*, 42, 7-49.

Gray, W. D., & Atwood, M. E. (in press). Transfer, Adaptation, & Use of Intelligent Tutoring Technology: The Case of Grace. In M. Farr and J. Psotka (Eds), *Intelligent computer tutors: Real world applications*. New York, NY: Taylor and Francis.

Gray, W. D., Radlinski, R. E., Burns, B., Sabnani, H., Mamone, S. J., Morch, A., & Atwood, M. E. (submitted). Reaching students through intelligent tutoring technology: A trial of ITS based Remediation.

Address all correspondance to: Wayne D. Gray; NYNEX Science & Technology Center; 500 Westchester Avenue; White Plains, NY 10604.

Toward an Architecture for Instructional Evaluation

John M. Carroll, Mark K. Singley, and Mary Beth Rosson
IBM Watson Research Center
Box 704, Yorktown Heights, NY 10598

Abstract: We recruit the construct of psychological design rationale as a framework for organizing instructional evaluation—and in particular for addressing the problem of credit/blame assignment to artifacts that instantiate multiple theoretical models.

For several years, our group has been developing an integrated framework for psychological science and design in the domain of human-computer interaction (HCI). The springboard for our analysis is the observation that designed artifacts (hardware, software, applications, interfaces) can be interpreted as *theories*, as embodying myriad specific claims about their users and the circumstances of their use. For example, a programming environment can be seen as embodying a range of claims about what programmers know, what they do, what they experience, about the nature of programming tasks and the contexts within which these tasks are carried out.

In our approach, the psychological design rationale of an artifact-in-use is articulated in causal schemas relating properties of an artifact with specific psychological consequences, under the scope of a usage situation. Thus, for example, including animated demonstrations in a programming environment supports learning by exploration for situations in which the programmer wonders what sorts of projects to work on. Such claims can be grounded in general laws of psychology, specific user data, etc.

Most of our work to date has sought to expose and codify the psychological rationale of HCI artifacts in a way that can support more deliberate design analysis. We have demonstrated this possibility through case studies, largely focused on the design of instructional artifacts: training wheels interfaces (Carroll, Kellogg and Rosson, 1991), the View Matcher Smalltalk environment (Carroll and Rosson, 1991). In this paper we expand our interests to the evaluation problem endemic to design generally and instructional design in particular.

The Evaluation Problem

In his classic analysis of curriculum evaluation, Scriven (1967) noted that the performance of an instructional artifact can only be assessed against its *goals*. He stressed the need to characterize and evaluate curriculum goals, pointing out that "verbally espoused goals of a curriculum-maker are often not the implicit goals of his curriculum" (p. 54). However, Scriven worried that such "intrinsic" evaluation of implicit goals was both too demanding for evaluators and too removed from the empirical realities of instructional situations: "One of the charms of the pay-off type of evaluation is the lack of charm, indeed the messiness, of a thorough intrinsic evaluation" (p. 54).

Scriven proposed a compromise, or melding of intrinsic and pay-off evaluation in his notion of "mediated evaluation" (p. 55 f.). He suggested focussing intrinsic evaluation on what can be embodied in a comprehensive set of test questions directly applied in pay-off evaluation (p. 56), and on guiding credit/blame attribution in redesign (p. 60). Unfortunately, evaluation work still often falls short of this. It is typical to conduct a series of narrow but controlled experiments and/or to measure aggregate indicators (overall task times and error rates), without any systematic analysis of the implicit goals embodied in an instructional artifact or the ways in which these goals were embodied. The net is that one often evaluates an instructional object A as flawed or some other object B as better than A, but without any warrant for the criteria of evaluation or comparison, and without any means of attributing shortcomings or strengths in A or B to specific aspects of their design.

Psychological Design Rationale

Claim schemas provide an integration of Scriven's intrinsic and pay-off evaluation. They are intrinsic evaluations in that they enumerate specific properties of an artifact, but they go further in causally relating these properties to specific psychological consequences for learners in given instructional situations. As semi-formal representations of the psychology of learning embodied in an instructional artifact, they broadly support design analysis. But moreover, they immediately provide rich, empirical hypotheses for pay-off evaluation.

Table 1 presents some claims pertaining to the MITTS minimalist tutorial and tools set for Smalltalk programming (Rosson, Carroll and Bellamy, 1990). For example, MITTS embodies the claim that sparse procedures (specifying functions but not message names) force learners to reason causally about the situation, to reflect-in-action (Schon, 1983) about their programming project. The claim is qualified in that learners may not have access to enough information to come to correct conclusions, or may be uncomfortable bearing the responsibility for their learning. We have segregated the claims into those pertaining to the instructional process, that is MITTS's theory of the learning experience, and those pertaining to what is learned, the theory of the learning outcome.

The various claims embodied in MITTS are associated with scenarios of learning Smalltalk. That is, for a claim to be either true or false certain situations of use must obtain. Thus, to determine how sparse procedures impact learning, there must be some situation in which details need to be inferred. In Table 2, we list the signature scenarios for MITTS. For the theory of the learning experience, these are the scenarios suggested and described in the MITTS manual. For the theory of the learning outcome, they are the scenarios of Smalltalk programming we targeted in designing this instruction.

In addition to situating the claims analysis of an artifact, the scenario analysis itself provides further support for analysis and evaluation. For example, one can compare scenarios to compare educational goals for two artifacts; one can ask whether performing the learning experience scenarios could possibly support performance of the learning outcome scenarios. Such analyses can be reinforced by direct empirical study of how the signature scenarios of an artifact correspond to the obtained learner scenarios: can people do what the artifact "expects" them to do?

MITTS was designed to exemplify the minimalist instructional model (Carroll, 1990). This model is also an instructional artifact, albeit one more general and abstract than MITTS itself. We can construct causal schemas relating general properties of instructional materials and general types of learning scenarios to consequences for learners. In Table 3,

Table1

Psychological Claims Embodied in MITTS Minimalist Tutorial/Tool Set for Smalltalk

theory of learning experience	*theory of learning outcome*
• analyzing a blackjack card game orients and motivates learners by engaging prior knowledge (but may encourage context-specific learning)	• developing a single example in some depth promotes an understanding of the context in which Smalltalk programming skills are exercised (but the learner doesn't see components in a range of contexts)
• exploring the execution stack orients and motivates learners by engaging prior knowledge (but inappropriate procedural programming knowledge may be engaged)	• organizing methods and classes into projects provides a task-oriented rubric for reusing them by analogy (but may not cover the Smalltalk domain evenly and thoroughly)
• filtering classes and method code in the class hierarchy browser controls the initially-perceived complexity of Smalltalk (but experienced programmers may be frustrated by not having access to all classes and to code) (but this could cause negative transfer in the full system environment)	• emphasizing browsing and code comprehension skills equips the learner to search for and reuse code (but other skills like code generation may not be developed)
• specifying the function but not the message name in procedures forces learners to infer or retrieve the connections between goals and methods (but they might not have access to enough information to reason successfully) (but students may be anxious about bearing such responsibilities)	• specifying the function but not the message name in an exercise strengthens connections between goals and methods (but the methods generated and practiced may be non-optimal)
• working with the model first directs learners attention to the game's basic functionality (but may disappoint learners' expectations about Smalltalk applications)	• working with the model and view separately helps the programmer differentiate between an object and its presentation (but may encourage a programming style that proliferates objects and message-sends)
• permanent display and coordination of tools with the blackjack game focuses and motivates learners in the context of error diagnosis and recovery (but interpretation can be obstructed by the complexity of the display)	• self-directed exploration of BlackJack classes makes browsing skills more robust and accessible (but learners may pursue unproductive goals)

Table 2

Signature Scenarios for MITTS Minimalist Tutorial/Tool Set for Smalltalk

theory of learning experience	*theory of learning outcome*
• playing blackjack by creating and sending messages in a workspace	• using a workspace to create and exercise objects
• playing blackjack with a graphical interface	• writing and modifying methods
• exploring a deal request in the execution stack	• using the debugger to trace code execution
• investigating how the player takes a hit	• inspecting object states
• setting up the blackJack user interface	• finding and exploring classes and methods
• analyzing blackJack's hitMe method	
• enhancing the stay message	
• updating the dealer menu	
• directly modifying a card in the inspector	
• customizing the blackjack game window	

we have enumerated claims embodied by the minimalist model and by the "systems approach" model (Gagne and Briggs, 1979).

Table 3
Psychological Claims Embodied in Two Instructional Models

minimalist model claims	*systems model claims*
• working on a concrete task provides the learner with an appropriate framework for integrating and applying learning experiences (but instructional objectives may need to be inferred) (but there may be too many kinds of task settings to support)	• clearly specifying the immediate learning objective and the performance criteria to be used in assessing success helps the learner orient to the instructional situation and recognize when learning has occurred (but the learner may reject a learning objective that fails to be personally meaningful)
• working on a familiar task orients and motivates learners by engaging prior knowledge (but may encourage context-specific learning) (but inappropriate prior knowledge may be engaged)	• decomposing target objectives into component enabling objectives and training each of these in turn allows complex target objectives to be systematically built up from parts (but this organization may not facilitate retrieval and application in task settings)
• incorporating planning and acting throughout the learning experience helps orient the learner to applying knowledge and supports skill transfer (but increases task complexity)	• keeping directive instructions and learner practice in lock-step contiguity allows the learner to know exactly what to do and when (but this does not promote reflection or analysis on the part of the learner) (but learners must relinquish control)
• retrieval, elaboration, and inference-making engage and sustain learners' attention and make skills more robust and accessible (but learners might not have access to enough information to reason successfully) (but learners may be anxious about bearing such responsibilities)	• repetitive practice clarifies correct discriminations and smoothes performance (but complex tasks are not scripts)
• diagnosing and recovering from errors focuses and motivates learners and helps sharpen a concept of correct performance (but errors can be frustrating and disrupt task goals)	• making a pleasant, familiar experience contingent on a target performance reinforces the target performance (but this can undermine the intrinsic motivation for achievement)

Building psychological design rationales for both model artifacts (Table 3) and concrete artifacts (Table 2) makes explicit the sense in which instructional artifacts instantiate models: They generally draw on more than one model. In our framework, this is conceptualized as a multiple inheritance relation: claims embodied in the exemplar artifacts can be specialized from claims embodied in model artifacts. Table 3 shows that MITTS' claims specialize claims from the minimalist model and the system model.

For example, the sparse procedures claim specializes both of the models. It specializes the minimalist claim that retrieval, elaboration and inference-making focus attention and effort by binding these activities to the context of following procedures: for cases in which one needs to get a procedure just right, sheer discovery may not be efficient or pleasant. It also specializes the systems claim that lock-step instructions and practice keep learners oriented by focussing instruction on a functional level: sparse procedures provide guidance, but also require the learner to actively make sense.

This multiple inheritance mechanism is what allows our approach to evaluation to be theory-directed and yet seriously practical. Credit/blame attributions to model artifacts allow us to evaluate theories as we evaluate concrete instantiations of theories. However, we are not committed to or constrained by the laboratory-inspired idealization that complex, instructional artifacts *uniquely* instantiate theories. This allows us to make appropriate credit/blame attributions to multiple model artifacts.

An Example Evaluation

We recently conducted a formative evaluation as we redesigned MITTS for Smalltalk VPM. The evaluation architecture, described above, can be used to guide such work. For example, in one episode, two learners were setting up an environment of objects to help them analyze the hitMe method (Table 2). They read an instruction that said "As in Topic 1, create a new instance of BlackJack named Game1 and deal a new set of hands." This is a sparse procedure: it does not specify how to instantiate the BlackJack class or how to deal.

The learners flipped back through the manual to Topic 1 and searched for relevant details. They quickly retrieved the full procedure for creating a new instance, and carried it out. In the course of doing this, however, they seemed to lose track of the "deal" clause, and failed to deal new hands. In consequence, they encountered nil values for some objects as they continued working on the project. They recognized that something was wrong and began trying to diagnose the problem by taking stock of the objects they had and the messages each had received. After several minutes, they concluded (without ever re-reading the instructions) that a deal message had to be sent to the blackjack game. They did this and were able to continue.

This episode is the type of thing one regularly sees in formative evaluation: something went wrong; the obtained learning scenario in this case was different than the corresponding signature scenario for MITTS (analyzing the hitMe method). We can understand this episode in terms of the sparse procedures claim: The learners were induced to reason, they were able to retrieve the initialization details and eventually to infer the need to deal new hands (after several minutes work).

The episode qualifies the truth of both the sparse procedures claim and the signature scenarios it describes. The claim and its scenarios also provide a framework for further design analysis of MITTS, for hypothesizing redesign options and reasoning about their potential impacts on learner performance and experience. And the inheritance mechanism supports making credit/blame attributions and exploring design options in the *models* exemplified by MITTS, as well as in the artifact itself.

In the episode, the learners missed the deal clause in a sparse procedure. From the systems model perspective, this suggests a boundary condition for the lock-step claim: in functional-level procedures, the succession of steps has to be verified. Learners need protection from accidentally skipping a step. This, in turn, suggests the design strategy of increasing directiveness, detail and control; perhaps reminding the learner how to create instances, splitting out the deal clause on a separate line of the manual, or both.

Viewing the sparse procedures claim as a specialization of the minimalist model suggests boundary conditions for the retrieval, elaboration, and inference-making claim: instructions need to be justified in the task's goal structure. Learners must understand instructions as meaningful components of a task. This, in turn, suggests the design strategy of integrating the create and deal instructions: "Deal a new game."

We believe this episode militates more strongly for the minimalist approach in that its claim seems more clearly true here (and its associated qualifications more clearly false). However, our reasoning also depends on a global concern with providing a consistent instructional orientation for learners (thus, in this evaluation we were looking for justifications to make design moves away from minimalism, but assuming that other things being equal we would not do so).

Prospects

We have described an approach to integrating intrinsic and pay-off evaluation within the framework of psychological design rationale. Representing instructional artifacts in claim schemas relates goals and criteria to testable psychological consequences for specific usage situations. It focuses pay-off evaluation on a rich variety of psychological variables (versus simple aggregate indicators, or even fine-grained time and error tallies). Inheritance relations between claim schemas of exemplar artifacts and model artifacts allows theoretical abstraction and cumulation, and offers a way to escape the idealization that complex artifacts singularly instantiate monolithic theories. This work converges the "design space" with the "theory space"–making it feasible, indeed routine, to evaluate theoretical abstractions in the course of practical design evaluation.

Our immediate plans include developing the evaluation architecture to make a theory-directed summative evaluation of MITTS and a commercially-produced systems-style tutorial. We also want to extend our analysis to include radically different types of instructional artifacts; our future work will incorporate evaluation of the MoleHill intelligent tutor for Smalltalk (Singley and Carroll, 1990). Finally, we want to pursue the possibilities for better scientific cumulation that our approach seems to promise by building tools to manage and incorporate design rationales and evaluation results in a "science base" that we envision as a browsable knowledge base.

Acknowledgment: We are grateful to Steve Payne for comments on this paper.

References

Carroll, J.M. (1990). *The Nurnberg Funnel: Minimalist instruction for practical computer skill.* Cambridge: MIT Press.

Carroll, J.M., Kellogg, W.A., and Rosson, M.B. (1991). The task-artifact cycle. In J.M. Carroll (Ed.), *Designing interaction: Psychology at the human-computer interface.* New York: Cambridge University Press.

Carroll, J.M. and Rosson, M.B. (1991). Deliberated evolution: Stalking the View Matcher in design space. *Human-Computer Interaction, 6.*

Gagne, R.M. and Briggs, L.J. (1979). *Principles of instructional design.* New York: Holt, Rinehart and Winston.

Rosson, M.B. Carroll, J.M. and Bellamy, R.K.E. (1990). Smalltalk scaffolding: A case study of minimalist instruction. In J.C. Chew and J. Whiteside (Eds.) *Proceedings of CHI'90: Human Factors in Computing Systems.* New York: ACM, pp. 423-429.

Scriven, M. (1967). The methodology of evaluation. In R. Tyler, R. Gagne, and M. Scriven (Eds.), *Perspectives of curriculum evaluation.* Chicago: Rand McNally, pp. 39-83.

Schon, D.A. (1983). *The reflective practitioner: How professionals think in action.* New York: Basic Books.

Singley, M.K. and Carroll, J.M. (1990). Minimalist planning tools in an instructional system for Smalltalk. In D. Diaper, D. Gilmore, G. Cockton, and B. Shackel (Eds.), *Proceedings of Third IFIP Conference on Human-Computer Interaction - Interact'90.* Amsterdam: North-Holland, pp. 937-944.

Designing Environments that Invite Thinking and Innovation

The Cognition and Technology Group at Vanderbilt
Learning Technology Center
Peabody College at Vanderbilt University
Vanderbilt University, Box 45, Peabody, Nashville, TN 37240

Abstract: We will illustrate design principles for technology-based environments that support thinking and innovation by discussing our Jasper Woodbury videodisc-based problem solving series that is currently being used in nine different states. Funded in part by the James S. McDonnell Foundation, the National Science Foundation and Vanderbilt University, the series is designed to enhance the mathematical problem solving skills of fifth and sixth graders and to provide a context for exploring issues in social studies, science, literature and other areas.

Our discussion will focus on (a) a description of the problem solving series and the software that accompanies it; (b) data indicating the effects of the series on the problem solving abilities of fifth and sixth graders; (c) a description of the "Jasper Network" in the nine states and the feedback being received by it; (d) illustrations of how the design principles underlying the Jasper series can be extended to the design of curriculum in other content areas.

Direct Manipulation Authoring of Instruction in a Model-Based Graphical Environment

Lee D. Coller, Quentin A. Pizzini, James Wogulis,
Allen Munro, and Douglas M. Towne
Behavioral Technology Laboratories
University of Southern California
250 N. Harbor Drive, Suite 309, Redondo Beach, CA 90277

Abstract: RAPIDS II is a direct-manipulation environment in which authors compose interactive graphical models and instructional sessions based on those models. After a model has been constructed, an author can create instructional interactions in a direct fashion, by interacting with the model. For example, a procedure performance drill can be composed simply by going through the sequence of actions that students are to perform during the training session. At training time, the RAPIDS II run-time system provides automatic trial repetition and remediation in response to errors. The combination of direct manipulation authoring of training exercises and their automatic administration during training constitute a significant contribution to productivity in technical training development.

Introduction

Graphical model-based instruction presents to the learner a visual representation of a system that can be manipulated and observed. Ideally, the actions required of students are cognitively similar to the operations performed on the real system because the actions require the same knowledge to perform. A graphical model-based approach is appropriate for instructional content domains that make use of visual representations that respond to student actions. Such domains include the operation and maintenance of complex devices, the behavior of many physical, chemical, and biological systems, and many abstract process domains.

We use the term *device model* to refer to these dynamic interactive graphical models. In these device models, visual behavior is determined by the interaction of behavior rules of objects in the model. A well-defined device model can serve as the basis for an excellent discovery learning system. In addition, however, the device model approach can support the rapid development of authored instructional content by direct manipulation. A *direct manipulation* approach to the authoring of instructional content means that instructional materials are developed by the author interacting directly with the graphical model, in much the same way that students do.

Other Model-Based Graphical Environments for Instruction

STEAMER (Williams, Hollan, and Stevens, 1981; Hollan, Hutchins, and Weitzman, 1984) provided a direct-manipulation interface for students learning about a steam generation plant. It offered both a discovery world for the learner and a means for demonstrating effects by a knowledgeable instructor. It did not, however, provide authoring tools for the development and delivery of instruction to the learner. It also required that the device model be coded in a conventional programming language, rather than authored directly in the graphical environment. In spite of these limitations, the STEAMER application clearly showed the potential of basing instruction on a reactive graphical model of a complex system, and it showed the potential pedagogical benefits from attaching graphical objects to a device model.

IMTS (Towne and Munro, 1988, 1989, in press; Towne, Munro, Pizzini, Surmon, Coller, and Wogulis, 1991) provided tools for authoring device model representations of electrical and hydraulic systems. The model-authoring approach was limited since the builder was required to compose very low level models. Included in IMTS was a generic troubleshooting expert called *Profile* (Towne, 1984, 1986; Towne, Johnson, and Corwin, 1983), which trained students in fault isolation. While Profile provided a powerful and general pedagogical expert, IMTS only supported instruction in troubleshooting methodology.

Direct Manipulation Instructional Authoring Research

Spensley and Elsom-Cook (1989) have developed a system called Dominie that supports the authoring of instruction by direct manipulation in an authored graphical environment. The visual environment, however, is inherently passive. Because there are no defined behaviors for graphical elements, any consequences of user actions must be explicitly authored along with other instructional materials.

RAPIDS (Towne and Munro, 1990; Munro and Pizzini, 1990) is a descendent of IMTS that supports direct manipulation authoring of instructional content in the context of authored interactive graphical models. Its authoring environment supports the development of graphical objects with rule-based behaviors, as well as the development of instructional content and curriculum design.

Direct Manipulation Authoring in RAPIDS

RAPIDS was designed to:

- provide a realistic portrayal of the system to be learned, including the direct-manipulation of device models by students and instructors, accurate graphical portrayal of responses to user actions, and accurate portrayal of real-time and continuous effects;
- support a componential approach to object modeling--this modular approach helps to improve the correctness, reusability, and maintainability of domain models;
- provide support for the simulation of device failures at a component level of representation;
- allow subject matter experts to demonstrate and explain procedures and processes by operating on the model, to control the administration of instruction via a high-level plan produced using a graphical interface without requiring detailed curriculum control programming, and to separate the technical content from the instructional methods so that the delivery of content could be varied by automated processes.

The Device Model

The RAPIDS modeling environment consists of objects that contain attributes and rules for describing the relationships among attributes. Objects may have several graphical appearances, each one corresponding to a different state of the object. A special attribute called *state* is used for maintaining states of objects. For some objects, state can be represented by a single appearance that is rotated or repositioned. RAPIDS is designed as a real-time modeling environment, thus during instruction real-time constraints need to be maintained. This includes the time at which an action must be performed and the spacing between actions on the model.

Device models are produced through an interface that allows users to graphically construct models and to define attributes that maintain information used to determine their operation and rules that define their operation. Authors first create prototype (generic) objects, and instantiate them to produce the device model. Additional rules must be provided to describe how various objects interrelate. To facilitate authoring, some rules that describe relationships between objects are automatically created. Figure 1 shows the environment used for the direct manipulation authoring of instruction and includes a device model.

Creating Instruction by Direct Manipulation

Once the device model is built, the author can use it to build modules of instruction. For much of what goes into these modules, the author manipulates the device model in the way that the student will subsequently be expected to manipulate it. As the author performs an action, RAPIDS records what action was performed. The device model responds to this action. For example, if the author changes a switch setting, RAPIDS records the action, and the model reacts to the new setting, propagating effects appropriately throughout the system. Consequently, the authors and students are always presented with accurate graphical representations of the system.

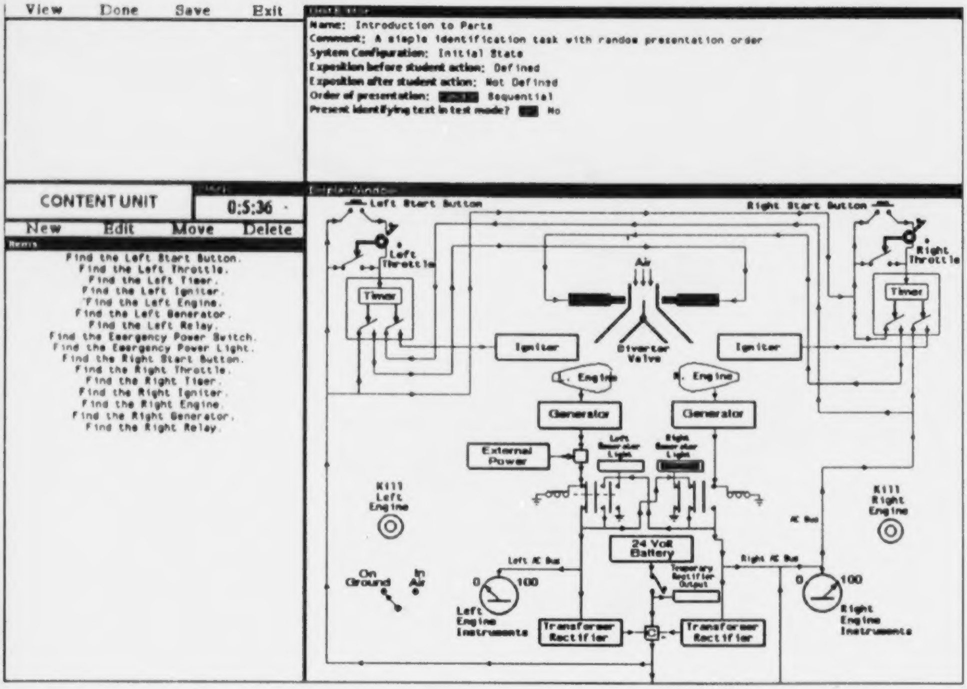
Actions

Author actions that can change the device model are:

- set switches (user-manipulable graphical objects);
- insert failed components (model objects that behave like faulty devices);
- replace failed components with normal ones;
- connect test equipment and make readings;
- install a complete system configuration, possibly including failed components; a system configuration is a "snapshot" of the device model at a given moment;
- get the system into a particular configuration that the student will be required to match; typically the device model is put into one configuration, and the student must get it into the other.

Authoring actions that do not change the device model are:

- identify things in the device model–individual objects, sets of objects, regions of interest;
- find indicators and observe their states or values.


View
Done
Save
Exit
CONTENT UNIT
0:5:36
New
Edit
Move
Delete
Left Start Button
Right Start Button
Left Throttle
Right Throttle
Timer
Timer
Air
Igniter
Igniter
Diverter Valve
L. Engine
R. Engine
Generator
Generator
External Power
Left Generator Light
Right Generator Light
Kill Left Engine
Kill Right Engine
AC Bus
24 Volt Battery
Temporary Rectifier Output
Left AC Bus
Right AC Bus
On Ground
In Air
0
100
0
100
Left Engine Instruments
Right Engine Instruments
Transformer Rectifier
Transformer Rectifier

Name: Introduction to Parts
Comment: A simple identification task with random presentation order
System Configuration: Initial State
Exposition before student action: Defined
Exposition after student action: Not Defined
Order of presentation: ▇ Sequential
Present identifying text in test mode? ▇ No

Find the Left Start Button.
Find the Left Throttle.
Find the Left Timer.
Find the Left Igniter.
Find the Left Engine.
Find the Left Generator.
Find the Left Relay.
Find the Emergency Power Switch.
Find the Emergency Power Light.
Find the Right Start Button.
Find the Right Throttle.
Find the Right Timer.
Find the Right Igniter.
Find the Right Engine.
Find the Right Generator.
Find the Right Relay.

Figure 1. A RAPIDS direct manipulation authoring environment

Authoring actions that do not involve interaction with the device model are:

- build menus for multiple choice questions of various types;
- allow the student freely to operate the device model, without limitations.

As the author performs an action in the graphical model, text is automatically generated that will be presented to the student as a prompt for the action that is to be performed. For example, "Set the Main Power Switch to On." Authors may always modify the text, but in many cases the generated text is sufficient. In other cases RAPIDS is unable to determine what is required of the student; therefore, authors must provide the text.

By collecting the appropriate actions into an instructional module, the author can build various types of exercises. An identification drill would consist of a sequence of identification actions or multiple choice menus. A procedural drill would consist of a sequence of switch settings. A fault isolation drill would consist of a sequence of indicator observations and test equipment readings.

BEST COPY AVAILABLE

Exposition

In addition to defining the actions described above, authors can compose expository material to be presented to students either before or after the action is initiated. Exposition can consist of sequences built from text, highlighting of objects or regions of the device model, changing the device model scene, or playing a videodisc segment. Building exposition never causes changes to the device model, but some elements do involve interacting with it, such as selecting an object to be highlighted. Authors may allow students to control the pace of presentation through the use of student mouse clicks, or they can control it themselves by placing arbitrary length pauses in the exposition.

Building Instruction

As authors build instructional modules, they will have a clear idea of what students will see and do since the student situation will be little different from what the authors have seen and done. For example, the author builds a switch setting action by manipulating a switch; the student manipulates the switch in the same manner. The author builds an identification action by clicking on an object; the student responds to an identification prompt by clicking on an object.

The author has considerable flexibility in deciding what goes into an instructional module. Typically a module will include some preliminary exposition for the module as a whole, the set of action units, and finally summary exposition for the module as a whole. Each action unit may include exposition preliminary to that particular action, then the action itself, then exposition that discusses what might have been observed or learned as a result of that action.

As students attempt to match the actions specified by the author, RAPIDS monitors whether they have responded appropriately. If a particular action is incorrect, the student is so advised and given a second chance. If a second failure occurs, RAPIDS will perform the action correctly and remediate, telling them what they actually did. If the student is operating in test mode, RAPIDS will only perform the action if the action affects the device model's state.

Instructional Plan

Instructional plans are used to control the delivery of instruction to students according to a general scheme of instruction specified by the author. Plans are hierarchies of topics and subtopics (called *Organizational Units* in previous papers), with leaf nodes representing instances of content units (called *Content Unit Nodes*). The term *unit* refers to both topics and content unit nodes. This hierarchy determines how a course is presented. For the most part, instructional content is authored separately from plans; however, topics can include expositions that provide for introductions to and summaries of topics.

Plans are created using a visual graph editor. By selecting nodes and choosing various menu options, authors can add and delete nodes and change the ordering of the children of a node. This interface allows authors to visually manipulate the structure of the plan so that it provides a natural breakdown of the various topics to be presented.

Authors have the ability to control the level of mastery students are required to achieve for each topic and subtopic. Content units are scored by measuring the percentage of correct responses and the amount of time spent completing the content unit. Topics are scored on the basis of the successfulness of their suptopics.

An example of an instructional plan is shown in Figure 2. This is a plan that was developed for instruction of a simple jet engine starting procedure (Kieras, 1988). In this plan, content unit nodes are represented by dashed nodes and topic units by solid nodes. Assuming the student performed everything successfully, the order of presentation of the content units in the plan would be from top to bottom, doing "Intro to Parts" first and concluding with "Dead in Air Drill."

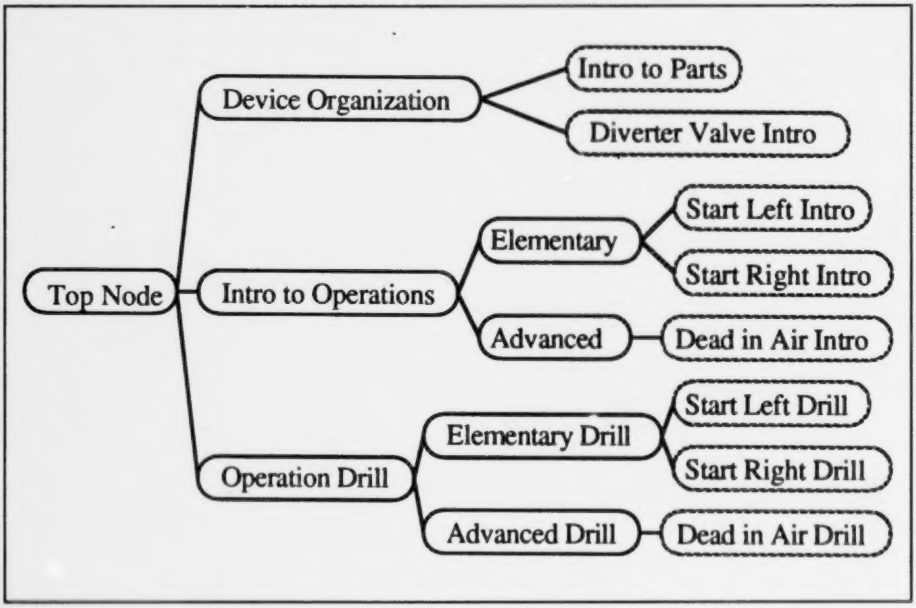

Top Node
Device Organization
Intro to Parts
Diverter Valve Intro
Intro to Operations
Elementary
Start Left Intro
Start Right Intro
Advanced
Dead in Air Intro
Operation Drill
Elementary Drill
Start Left Drill
Start Right Drill
Advanced Drill
Dead in Air Drill

Figure 2. Instructional Plan

Instructional Modes

There are three modes in which instructional content can be presented: *Instruct*, *Drill*, and *Test*. This presentation mode is designated by authors when constructing instructional plans.

Instruct mode is the initial learning mode for the student. In this mode all instructional material is demonstrated; the only action taken by students is clicking the mouse to pace the presentation. Operations that would normally require students to manipulate the device model are demonstrated by RAPIDS.

In drill mode, as in instruct mode, all instructional material is presented. However, instead of merely observing, students are required to perform manipulations in the same manner that authors did when constructing the unit. Incorrect manipulations result in the presentation of corrective text and a demonstration of the correct actions.

In test mode, the system tests students on their knowledge of material; instructional material (expositions) are not presented in this mode. Students are only presented enough information to prompt them to perform the required actions and answer the required questions. The feedback generated only indicates if the response was correct or incorrect.

Incorrect responses that affect the device model state are corrected because a correctly configured model may be required to complete the unit.

Conclusion

RAPIDS provides an interactive environment in which authors can build active graphical device models and instruction based on those models. The author builds instructional content units by directly manipulating the device model to demonstrate procedures or tasks to be taught to the student. Much of the instructional interaction with the student is automatically generated from the author's interaction with the model but can be further elaborated by the author to include other textual, graphic or videodisc presentations. Higher level instructional plans that control the presentation of content units to the student are built graphically and may be reused to organize course material for different domains. Direct manipulation authoring of device models and instruction in RAPIDS II provides authors with a simple and powerful means of creating, modifying and maintaining instruction in a model-based graphical environment.

Current Work

Currently we are working on the RIDES system, a successor to RAPIDS that will run in the Microsoft Windows 3.0 environment. In RIDES we are extending the instructional features explored in RAPIDS to include a more integrated instructional and simulation environment and to allow for the automatic generation of instruction that can be derived from the underlying device model.

IMTS was developed under funding from the Office of Naval Research and from the Navy Personnel Research and Development Center under contracts N00014-85-C-0040 and N00014-87-C-0489. Development of RAPIDS was funded by the Air Force Human Resources Laboratory, under RICIS Research Activity No. ET 13 (NASA Cooperative Agreement NCC9-16).

Acknowledgements: David S. Surmon collaborated in the design and development of IMTS and RAPIDS. Mark C. Johnson aided in the design of the *Profile* troubleshooting expert of IMTS.

References

Hollan, J. D., Hutchins, E. L., and Weitzman, L. 1984. STEAMER: an interactive inspectable simulation-based training system, *The AI Magazine*, 1984, 2.

Kieras, D.E. (1988). What mental model should be taught: Choosing instructional content for complex engineered systems. In J. Psotka, L.D. Massey & S. Mutter (Eds.), *Intelligent Tutoring Systems: Lessons Learned*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Munro, A. & Pizzini, Q. A. Direct manipulation of object behavior in an interactive graphical modeling environment. *The proceedings of the 1990 IEEE international conference on systems, man, and cybernetics*, Los Angeles: IEEE, 1990.

Spensley, F. and Elsom-Cook, M. Generating domain representations for ITS. In D. Bierman, J. Breuker, & J. Sandberg (Eds.), *The Proceedings of the Fourth International Conference on Artificial Intelligence and Education*. Amsterdam: IOS, 1989, 276-280.

Towne, D. M. A generalized model of fault-isolation performance. In *Proceedings, Artificial Intelligence in Maintenance: Joint Services Workshop*, 1984.

Towne, D. M. A generic expert diagnostician. In *The Proceedings of the Air Force Workshop on Artificial Intelligence Applications for Integrated Diagnostics*, 1986.

Towne, D. M., Johnson, M. C., and Corwin, W. H. *A performance-based technique for assessing equipment maintainability.* Los Angeles, CA: Behavioral Technology Laboratories, University of Southern California, Report No. TR-102, 1986.

Towne, D. M. & Munro, A. The intelligent maintenance training system. In J. Psotka, L. D. Massey & S. A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc., 1988.

Towne, D. M. & Munro, A. Artificial intelligence in training diagnostic skills. In D. Bierman, J. Breuker, & J. Sandberg (Eds.), *The Proceedings of the Fourth International Conference on Artificial Intelligence and Education*. Amsterdam: IOS, 1989, 291-297.

Towne, D. M., & Munro, A. Intelligent instruction from deep models. In *Advances in Human Factors Research on Man/Computer Interactions*: Proceedings of the conference of the Human Factors Society, 1990, Nashville, Tennessee: The Human Factors Society.

Towne, D. M. & Munro, A. Two approaches to simulation composition for training. In J. Psotka and M. Farr (Eds.), *Intelligent instruction by computer: from theory to practice*. London: Taylor and Francis, in press a.

Towne, D. M., Munro, A., Pizzini, Q. A., Surmon, D. S., Coller, L. D., & Wogulis, J. L. Model-building tools for simulation-based training. *Interactive Learning Environments*, 1990, 1, 33-50.

Williams, M. D., Hollan, J. D., and Stevens, A. L. An overview of STEAMER: an advanced computer-assisted instruction system for propulsion engineering. *Behavior Research Methods and Instrumentation*, 1981, 13, 85-90.

Design Issues for Interactive Learning Environments

Allan Collins
Bolt, Beranek and Newman, Inc.
10 Moulton St., Cambridge, MA 02138

Abstract: The talk discusses the design of interactive learning environments in terms of 25 issues that a designer should consider in building any system. The design issues range over the purposes, content, teaching methods, and sociology of the learning environment.

An Architecture for Apprenticeship: Collaboration with an Intelligent Tutoring System for Qualitative Electrical Troubleshooting

Eric B. W. Cooper
School of Education
Stanford University, Stanford, CA 94309

Abstract: QUEST is an intelligent tutoring system for basic electricity. It combines a graphical circuit editor and a qualitative simulation engine. The engine can provide causal explanations for its actions. The engine is capable of operating in two modes: *user control* and *demo mode*. Under user control it tracks the user's progress through the simulation. Under demo mode it performs the next step(s) confronting the student. The user can toggle between modes.

There is a problem driver that presents sequences of problems that highlight the aspects of circuit behavior that the student should focus on. This allows the system to direct the student's progression of mental models of electricity and simple circuit behavior.

I am hopeful that this architecture which allows the user to collaborate with the system in problem solving will lead to further research of cognitive apprenticeship between human learners and moderately intelligent computer tutors. This paper deals with the lessons learned in the design and implementation of QUEST.

Introduction

Intelligent tutoring systems and discovery-based microworlds span quite a range of pedagogical approaches. Some are capable of coaching by providing hints or critical feedback. Many can tell when a particular user response is incorrect, and can give a correct answer, while others provide an environment for open-ended exploration. QUEST, the system described here, is innovative in that it attempts to foster a master-apprentice relationship with the learner in the domain of basic electricity. In doing this, it poses problems to the learner and watches the learner's progress. QUEST critiques the learner's actions. If she so wishes, the learner can ask for additional comments or explanations. Furthermore, the learner can at any time ask the system to take over and demonstrate how the problem might be solved.

The learner is ultimately the one in control. She decides whether to solve the problem with minimal commentary from the system, or whether to solicit additional comments, or whether to let the system show how it would go about solving the problem at hand. Of course, the learner can freely intersperse all of these approaches.

This paper describes some of the architectural decisions that were made in investigating the degree to which the traditional master-apprentice learning environment can be realized by intelligent tutoring systems. For studies of this mode of teaching and learning, and its juxtaposition with the modes most prevalent in the classrooms of today see Lave (1987) and Collins, Brown and Newman (1989). In the context of teaching basic concepts of electrical circuits, we explored the reintroduction of apprentice-style learning.

The master electrician in QUEST is moderately intelligent at best. Unlike a human it does very little by way of modeling the prentice's understanding over time. How we have dealt with these limitations will also be discussed in this paper. The system described here raises many interesting questions and certainly there are as many issues that should be further investigated.

Overview of QUEST

At its core QUEST (Qualitative Understanding of Electrical System Troubleshooting) is a graphic circuit editor and a qualitative simulation engine. The engine is able to articulate reasons for its actions. The engine is capable of either tracking the learner's progress through the simulation, or of performing the simulation and talking (via a DECTalk unit) about this process.

Simulation of circuit behavior is composed of four distinct phases:

1) parsing the circuit into series and parallel paths (and determining conductivities of these paths),
2) assigning voltage (and/or current) information to paths and parts of paths,
3) evaluating device states, and
4) deciding whether or not the circuit has reached a steady state.

The core system can simulate and explain the behavior of most of the circuits that one would care to construct. (The devices that we have implemented are batteries, switches, wires, bulbs, resistors, high impedance test lights and NPN transistors.) In addition, we have built a problem driver that allows us to design sequences of problems that highlight the aspects of circuit behavior that we wish the student to focus on. This notion of supporting the learner through a progression of mental models of electrical circuitry behavior is covered extensively by White and Frederiksen (1987).

The learner typically interacts with the system in the context of a particular problem to be solved. Problem types include parsing problems (designed to introduce the learner to the notion of circuit parsing), prediction problems ("Will light bulb 1 be on?"), troubleshooting problems ("Find the part that is faulted open."), and design problems ("Using <parts list> build a circuit that will satisfy <some constraints>."). In the case of most problems the learner is presented with a circuit and various "tools" (e.g., ability to insert test lights, flip switches, etc.). These tools vary between problem types.

For example, a parsing problem might show the learner the circuit in Figure 1 and ask her to simulate its behavior. In doing this the learner will need to analyze the circuit's structure in terms of series and parallel paths. It is worth noting that the learner is allowed to parse this circuit non-optimally. So, while the system would parse it into a parallel path with SW2 and BULB2 which is, in turn, in series with the rest of the components, the learner might choose to parse BULB1, SW1, and BULB3 into a series path first, then find the parallel path, and then put these two paths into a series path. If the learner tries to say that BULB2 and BULB1 are in series, then she will receive appropriate feedback.

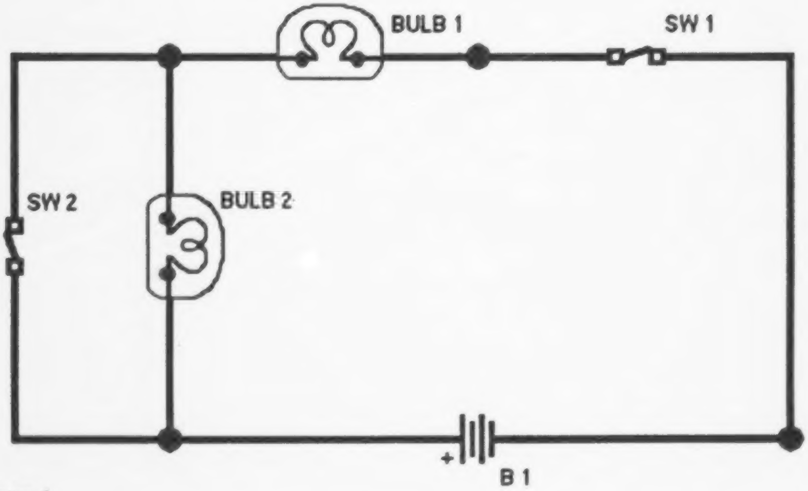


Figure 1.

Figure 1 could also be used by a prediction problem that asked, "Will BULB2 be on or off?" The learner can answer without going through the entire simulation. Figure 2 could be used for a troubleshooting problem where the instruction is simple, "Find the fault." In this case, there are a couple of possibilities. For example, it could be that there is a short to ground somewhere between BULB1 and BULB3.

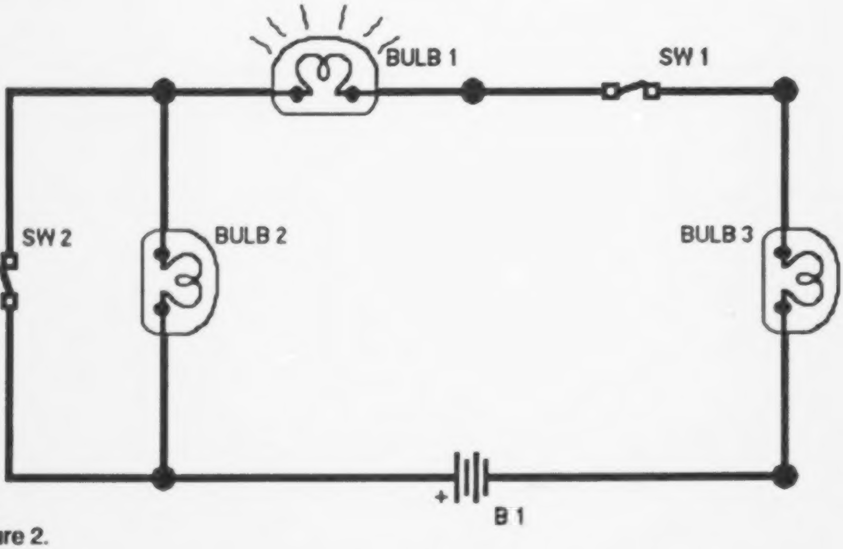


Figure 2.

Finally, a circuit like that in Figure 3 might be the response to a design problem that gave the learner one battery, one light bulb, and one switch, and said, "Design a circuit such that the bulb goes on when the switch is open and off when the switch is closed."

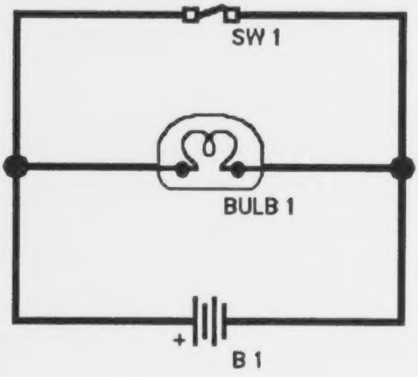


Figure 3.

Apprenticeship

Our attempts to incorporate an apprenticeship model are particularly visible in three features of the system. First, the learner can (at any time) switch between *user control* and *demo mode*. In demo mode the system performs the task at hand and articulates its actions as it goes. Under user control the learner performs actions and the system critiques these actions. The system says different things about the "same" action depending on who performed it (the system or the learner).

Second, in the spirit of Burton and Brown's WEST (1979), there are levels of commentary and the learner can regulate the amount of advice that she receives. Advice is often given tersely at first, but the learner can ask for a more in-depth reason for a certain action or comment. For example, the learner may have said that a parallel path in Figure 1 is conductive and resistive. The system will first tell the learner that, in fact, the path is purely conductive. If the learner asks for "more help," then the system replies that if a parallel path has a single part that is purely conductive, then the whole path is also purely conductive. Asked for still more elaboration, the system will highlight the closed switch (SW2) and point out that it is purely conductive and, hence, acts as a short across the bulb (BULB2).

Third, in lieu of an expert teacher who monitors the prentice's progress and constructs meaningful and appropriate tasks in response, we have implemented a *learning browser*. This is a graph or *lattice* of groups of related problems. Each node in this lattice contains a set of problems that are designed to explicate a particular topic. For example, one topic might deal with recognizing a circuit, e.g., realizing that a single wire from one port of a battery to one port of bulb is not enough for the bulb to light up. Another topic might focus on parallel paths and the notion of a short across a resistive component. The problem groupings, or levels, are arranged so that they build on previous levels and, perhaps, draw attention to possible overgeneralizations that the learner may have made. Currently the learning browser is quite linear, but this need not be the case. For the most part the learner can navigate through the browser at will. However, it may sometimes be desirable that a specific level only be accessible after certain prerequisite topics have been covered.

Architecture

The domain of circuits and electrical troubleshooting is fairly constrained in the actions that are allowable and the order in which actions can be taken. Most importantly there is a certain temporal ordering of events that needs be respected. Within these constraints, however, there is still a great deal of freedom. We have already mentioned the types of problems that QUEST provides, and each problem type can be applied to the very large number of possible circuits. This next section discusses the underlying control structure that takes advantage of the necessary constraints to maximize the flexibility of the system in responding to the learner.

Underlying the QUEST simulation engine is a *constraint dependency* structure. For each action there is an expression of predicates that must be satisfied before that action can be performed. This expression describes preconditions for the part that is being acted upon (if appropriate) as well as preconditions for all the parts in the circuit and for the circuit as a whole. For example, in order to start assigning voltages and currents one must have finished analyzing the circuit (i.e., parsing it into series and parallel paths). In other words, the precondition for entering into the voltage and current propagation phase of the simulation is that the circuit must have already been analyzed into paths and loops.

The constraint dependency structure could easily be configured so that currents could only be assigned to parts after all parts had had their voltages assigned; this would force the learner to focus on the concept of voltage rather than that of current. In this case the action of assigning current would have as its precondition that all parts satisfied the predicate VoltageAssigned? Alternatively, we could configure the system so that voltage could only be assigned to a part that had already had its current assigned. In this case the action of assigning voltage would have the precondition that the part in question satisfy the predicate CurrentAssigned? Notice that these are not simply reversals of each other. The first demands that all voltages be assigned before any currents can be assigned, the second merely requires that a given part has had its current specified before it voltage can be specified.

Under user control the constraint dependency structure is used simply to determine whether or not a particular action is permissible. If not, then the *explanation generator* takes over and gives the learner a reason why the action can not be performed. Usually this kind of advice is accompanied by suggestions about what the learner could do first in order to perform the specified action. The explanation generator is something like the *speakers* in WEST (Burton & Brown, 1979). Unfortunately, it falls outside the scope of this paper.

In demo mode this control structure actually drives the simulation engine. When simulating, the top goal is to "exit simulate" (i.e., finish simulating), but the preconditions for doing that involve making sure that the circuit has been analyzed, that voltages have been assigned, etc. Each precondition that has not yet been satisfied pushes the next necessary action onto the goal stack. Although we have not taken advantage of this, it is possible that each such sub-goal could be explained with respect to the larger goals that it will help realize. The system might say, "We want to simulate the circuit, but in order to do this we must first analyze its paths and loops." Later the system might say, "We want to simulate the circuit. We have analyzed it. Now let's propagate voltages." In other words, the system could be made to talk in terms of higher order goals.

There is a homunculus that monitors the learner under user control and drives the simulation in demo mode. The homunculus' job is to take the top goal off the goal stack and, see if its preconditions have been satisfied. If the preconditions have been satisfied then the homunculus performs the action. Otherwise, depending on the control mode, either it get the explanation generator to explain why the action could not be performed or it pushes the goal back onto the goal stack along with another action that will help satisfy the aforementioned preconditions. This bottleneck is what allows control to be handed off back and forth between the learner and the system.

Whether it is the learner or the system doing the action the screen activity looks much the same. Under demo mode the same menus still come up and the same items are selected that the learner might have selected under user control. This way the learner receives coaching on the subject matter as well as on how to interact with the system.

In addition to user control and demo mode there is a *silent mode*. This is useful for example in prediction problems where it is necessary to simulate circuit behavior, but stepping through the entire circuit analysis phase can become rather tedious. Prediction problems configure the system so that analyzing the circuit structure can be done either silently or interactively at the learner's discretion. Similarly, stepping through the entire simulation is not appropriate when working on troubleshooting problems, so it is done silently.

Conclusion

So far our master electrician only knows about circuit simulation: the analysis of paths, the assignment of conductivities, the propagation of voltages and currents, the evaluation of device states, and the recognition of a steady state. The next step is to extend this to the tasks of troubleshooting and of circuit design. As with the case mentioned above of circuit parsing there are usually several ways to go about troubleshooting a specific fault. It is important that the master be able to pick up where the learner left off and, following as much as possible the learner's strategy, continue working on isolating the fault.

Apprentice-style learning has become increasingly rare in our mass-production education system. This research begins to investigate the role that computers might play in turning this trend around. We have relied on the natural intelligence of the learner as much as possible.

Acknowledgements: I am particularly grateful to John Frederiksen and Barbara White for their support, encouragement, and lengthy discussions. The original version of QUEST was programmed by Frank Ritter, and we spent many hours discussing implementation issues and their ramifications. I would also like to thank Allan Collins, Bruce Roberts, and Eric Nyberg. This research was carried out while I was at BBN Labs and was funded by the Army Research Institute, under contract MDA-903-87-C-0545.

References

Burton, R. R. and Brown, J. S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies* 11: 5 - 24.

Collins, A., Brown, J. S., and Newman, S. (1989). Cognitive apprenticeship: teaching students the craft of reading, writing, and mathematics. In L. B. Resnick (Ed.), *Knowing, learning, and instruction: Essays in honor of Robert Glazer*. Hillsdale, NJ: Erlbaum.

Greeno, J. G. (1988). *Situations, Mental Models, and Generative Knowledge.* (Report No. IRL88-0005) Palo Alto, CA: Institute for Research on Learning.

Lave, J. (1987). *Tailored learning: Apprenticeship and everyday practice among craftsmen in West Africa*. Unpublished manuscript.

White, B. Y. and Frederiksen, J. R. (1987). *Causal Model Progressions as a Foundation for Intelligent Learning Environments*. (BBN Report No.6686) Cambridge, MA: BBN Labs.

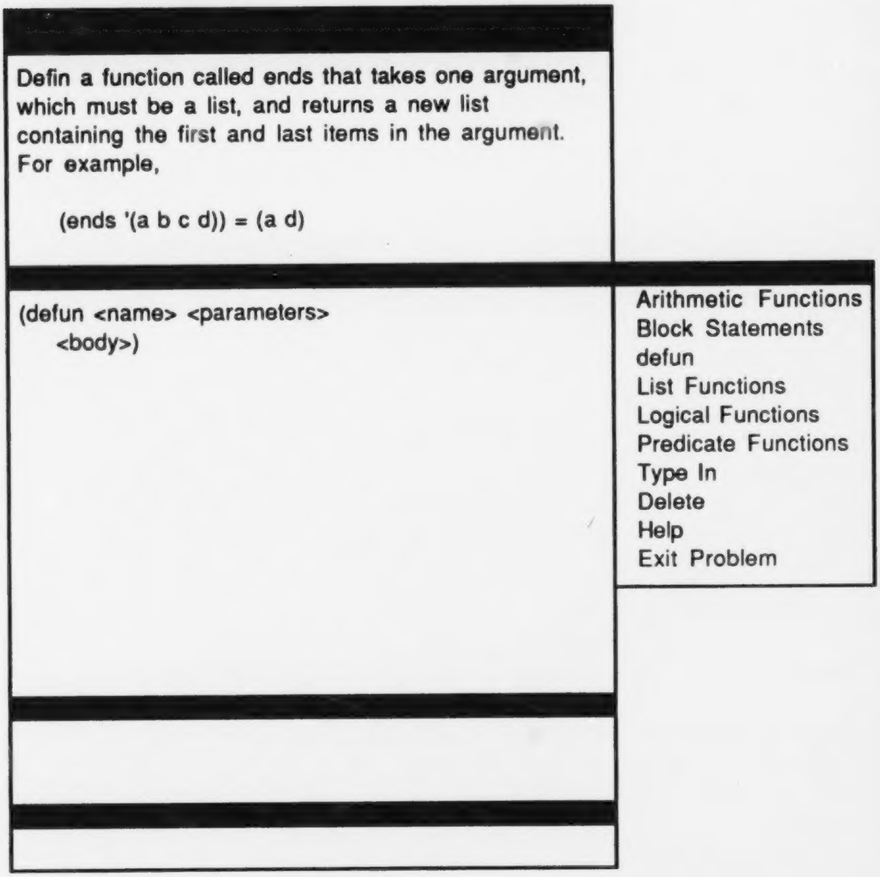
Menu Selection vs. Typing: Effects on Learning in an Intelligent Programming Tutor

Albert T. Corbett, John R. Anderson, and Jon M. Fincham
Department of Psychology
Carnegie Mellon University, Pittsburgh, PA USA 15213

Abstract: A study is reported comparing a menu-driven and type-in interface for an intelligent tutoring system that assists students with Lisp programming exercises. In the menu-driven condition, students selected code symbols from a two-level hierarchical menu system. In the type-in condition students typed each symbol in their programs. In either case, the tutor traced the students' solutions symbol-by-symbol by means of a production system student model and kept the students on a correct solution path. Contrary to our expectations, menu-driven code entry impeded learning. Students in the menu condition made more mistakes in completing the tutor exercises and performed substantially worse on post tests. Students in the menu condition generally had a more positive opinion of the tutor, although not if they switched to it from a type-in interface.

This study compares a menu-driven and a type-in interface in a practice environment for programming exercises. Students in this study worked through an introduction to the Lisp programming language. They read six chapters of a Lisp programming text and completed exercises with the assistance of the CMU General Programming Languages Tutor (Anderson, Corbett, Fincham, Hoffman and Pelletier, in press). In each exercise, this tutor presents a description of a short program to write and as the student enters a solution, the tutor monitors the student's performance symbol-by-symbol, providing feedback on errors and providing a correct action if the student appears to be floundering.

Figure 1 presents a snapshot of the screen as the student begins an exercise with the tutor. The problem description appears in the top window and the student's solution appears in the code window immediately below. Messages from the tutor appear in the third window. Figure 1 depicts the menu-driven version of the tutor. As can be seen the menu appears immediately to the right of the code window. In this snapshot, the student has just selected **defun** from this menu and the tutor has expanded a template for this operator in the code window. The template includes a set of parentheses and three goal nodes in angle brackets. These nodes are not themselves Lisp code, but serve as goal reminders which the student replaces with Lisp code. Note that the tutor employs a hierarchical menu system. With the exception of **defun**, no code symbols appear in the top level menu. Rather, operators are organized by categories. To enter the arithmetic operator +, for example, the student selects the entry **Arithmetic Functions**. Then the student selects + from the menu that appears. Certain symbols, e.g., numbers and variables must be typed. In this case, the student selects **Type In** from the top level menu, a more specific category from the resulting submenu, then types the symbol. In the type-in version, students type all code symbols in their solutions without any menu operations. If the student makes a mistake, the tutor interrupts immediately to notify the student. The tutor allows the student to try again, and does not provide any explanatory feedback. At any time, however, the student can ask for help at a goal node.



Defin a function called ends that takes one argument, which must be a list, and returns a new list containing the first and last items in the argument. For example,

```
(ends '(a b c d)) = (a d)
```

```
(defun <name> <parameters>
    <body>)
```

Arithmetic Functions
Block Statements
defun
List Functions
Logical Functions
Predicate Functions
Type In
Delete
Help
Exit Problem

Figure 1. A snapshot of the screen as the student begins an exercise with the programming tutor.

The tutor employs a model tracing paradigm to provide assistance to students (Anderson and Reiser, 1985; Corbett, Anderson and Patterson, 1990). The tutor contains a production-system model of the knowledge students need to complete the exercises in the curriculum. This student model allows the tutor to solve the exercises step-by-step along with the student and to provide help if a student makes a mistake. The architecture of the program is similar to the CMU Lisp Tutor, which has been shown to be an effective practice environment (Anderson and Reiser, 1985; Corbett and Anderson, 1991). The architecture of the newer tutor is more general than that of the Lisp Tutor, however, and can accept student models for a variety of languages. Currently, it also has a student model for Pascal and Prolog. (See also Gray and Atwood, in press, concerning a COBOL application).

The Lisp Tutor has a type-in interface, but in the course of developing the General Programming Languages Tutor we speculated that a mouse driven menu system might be more successful. Since menus are recognition based, they can make it easier for a novice to begin interacting with a system (Miller, 1988). We thought that at minimum, menus might reduce the time required to complete the tutor exercises. Beyond that, we speculated that menus might improve learning rate by reducing the cognitive load of entering code. To assess these possibilities, we implemented both interfaces for the tutor and compared them as described below.

Method

Subjects

Thirty-eight undergraduates participated in the experiment for pay. Math SAT and prior programming experience were controlled across groups.

Design

Students worked through 10 lessons with the tutor, completing 63 exercises. The first six lessons introduced new topics and were accompanied by text. Lessons 7-10 repeated the topics covered in lessons 3-6. Half the students used the menu-driven interface and half used the type-in interface for the first six lessons. For the remaining four lessons, nine students switched from the menu to type-in interface and ten switched from the type-in to menu interface. Students completed a paper and pencil test and filled out a questionnaire following the second, sixth and tenth lessons.

Results

Post tests

Results of the three paper and pencil post tests are displayed in Table 1. A strict accuracy measure is employed: percent of exercises completed without error. The upper half of the table displays the first two post test results for the two groups. As can be seen, students working with the type-in interface performed more accurately on both the first post test, ($F(1,36) = 5.27, p < .05$) and the second one ($F(1,36) = 6.77, p < .05$). Half the subjects switched interfaces prior to the final four lessons and post test. Thus, the menu and type-in columns are each divided into two columns at the bottom of Table 1. The two leftmost columns, respectively, represent students who started with menus and (1) stayed with menus or (2) switched to typing. Similarly, the remaining two columns represent students who started out typing and either stayed with typing or switched to menus. As can be seen, the three groups who used the type-in interface at some point performed similarly on the final post test, while the students who worked with menus throughout lagged behind. An analysis of variance of these results is marginally significant $F(3,34) = 2.22, p = .10$.

Table 1
Post Test Performance: Percent of Exercises Correct

	Menu		Type	
Test 1	38%		53%	
Test 2	11%		25%	
	Menu	Type	Type	Menu
Test 3	12%	24%	26%	30%

Tutor Exercises

Table 2 displays three measures of the students' performance in completing the tutor exercises. Again, the table is split, with the first six lessons at the top and the last four lessons at the bottom. The first measure in each half of the table is the mean time spent working on all exercises in a lesson. The two remaining measures describe performance at the level of individual goals and production firings. In these analyses, a production is said to have fired correctly when a student enters a correct code symbol on his/her first attempt at a goal. The two production-level measures are (a) mean time to enter a correct symbol and (b) the probability that a subject does not enter a correct symbol on his/her first attempt at a goal. (The first goal in each exercise is excluded from all analyses, since the time measures for this goal would include reading time for the problem description. Lesson 1 is excluded from the analyses, since the exercises involve at most 2-3 productions, one of which is automatically excluded.)

Table 2
On-line Performance Measures in Completing the Lessons:
Mean Time to Complete the Exercises in a Lesson, Mean Correct Production
Firing Time, Probability of an Error at a Goal

Lessons 2-6	Menu		Type	
Time/Lesson	24.8 min		22.1 min	
Time/Correct	11.6 sec		11.0 sec	
Errors	0.26%		0.18%	
Lessons 7-10	Menu	Type	Type	Menu
Time/Lesson	16.8 min	16.7 min	15.6 min	18.7 min
Time/Correct	7.7	7.6	7.7	9.2
Errors	0.23	0.24	0.19	0.21

There was no reliable difference across the two groups in time to complete the exercises nor in time for a correct production firing. However, students in the menu condition were more likely to make an error than students in the type-in condition, $F(1,36)=8.48$, $p<.01$.

There were no reliable differences among the time and accuracy measures for the final four lessons. The group that switched from typing to menus appears to have taken the longest to complete the exercises. If this is a real effect, it seems to be an interface rather than learning difference, since students in this group were as accurate as other groups, but took longer to enter a symbol.

Questionnaires

Each questionnaire contained seven to twelve questions that assessed students' self-monitoring of the learning process and how well they liked the tutor. Each of these questions employed a seven-point rating scale. The first two questionnaires asked how difficult the questions seemed and how well students had learned the material. On the second questionnaire, the exercises were rated more difficult in the menu condition than in the type-in condition (ratings = 5.5 vs 4.8, $F(1,36)=3.29$, $p<.08$). Otherwise, there were no differences. The questionnaires also asked if the tutor had helped the students complete the exercises more quickly or understand them better than than they might working on their own. On the first questionnaire, students in the menu condition believed more strongly than those in the type-in condition that the tutor had helped them complete the exercises faster (5.8 vs 5.2, $F(1,36)=3.12$, $p<.09$). Otherwise, there were no differences. In reality, students in the type-in condition finished the exercises as fast as faster. Finally, a variety of questions asked the students how much they liked the tutor, the code-entering facilities and the help facilities. There were no differences on the first questionnaire. Interestingly, on the second questionnaire, students in the menu condition liked the tutor better than those in the type-in condition (5.3 vs. 4.4, $F(1,36)=3.39$, $p<.07$), liked the code entering facilities better (4.7 vs. 4.0, $F(1,36)=3.01$, $p<.10$) and liked the help facilities better (5.2 vs 4.0, $F(1,36)=12.68$, $p<.01$). The help facilities were identical across the two conditions, but the overall pattern here suggests students in the menu condition are happier with the tutor at the end of the sixth lesson.

On the final questionnaire, there were reliable differences among groups on two questions. The group that switched from a type-in interface to a menu interface liked the tutor less than the other conditions (rating = 3.7 vs. 5.4, 5.7 and 5.2, $F(3,34)=4.19$, $p<.05$). The 3.7 rating is reliably smaller than each of the other three ratings. This same group found the interface harder to use than in the first six lessons, while the other three groups found their interface easier to use (rating = 5.0 vs. 3.0, 2.7 and 3.0, $F(3.34)=4.98$, $p<.01$). The 5.0 rating is reliably greater than each of the other three ratings. There were no other reliable differences among the groups in rating how hard the exercises were, how much they liked the coding and help facilities, or whether the tutor helped them understand the material better or more quickly.

Discussion

Students using a menu driven interface made more mistakes in completing the tutor exercises and performed substantially worse on the post tests. There are at least three possible reasons for the post test result. One is that the students in the menu condition merely suffered from interface incompatibility in the post test, since a paper-and-pencil test is recall-based. However, the fact that students in the menu condition were also making more mistakes while completing the tutor exercises suggests that there is more to the post test result than interface incompatibility. The evidence is that the menu-driven interface actually interfered with learning. We can imagine two reasons. First, menu selection may foster a trial-and-error approach to selecting code symbols at the expense of thinking more deeply about goals. Second, menu-selection, particularly with hierarchical menus, may represent a divided attention task, since students must repeatedly look away from the code, stop thinking in the exercise goal space, and think about where the desired symbol is in the menu. In effect, we may have increased cognitive load by introducing menus.

The questionnaire data are weakly suggestive concerning this issue. Initially, students working with menus believed more strongly that the tutor helped them finish quickly. If anything, students took longer in the menu condition, but students may have been confusing ease of problem solving with speed. If so, the ratings are consistent with the trial-and-error hypothesis. Dividing attention would be expected to increase perceived load. On the other hand, a trial-and-error approach in the face of uncertainty might be expected to reduce perceived load. By the second questionnaire, though, students in the menu condition perceive the exercises as more difficult, which appears to support the divided attention hypothesis. However, when exercises get hard, (which they begin to in lesson 3, judging from the post tests) students may also find it more burdensome to solve them with a trial-and-error method.

References

Anderson, J.R., Corbett, A.T., Fincham, J.M., Hoffman, D. and Pelletier, R. (in press). General principles for an intelligent tutoring architecture. In V. Shute & W. Regian (eds.) *Cognitive approaches to automated instruction.*

Anderson, J.R. and Reiser, B.J. (1985). The Lisp Tutor. *Byte, 10,* (4), 159-175.

Corbett, A.T. and Anderson, J.R. (1991). Feedback control and learning to program with the CMU Lisp Tutor. Talk presented at the Annual Meeting of the American Educational Research Association.

Corbett, A.T., Anderson, B.J. and Patterson, E.G. Student modelling and tutor flexibility in the Lisp Intelligent Tutoring System. In C. Frasson & G. Gauthier (eds.) *Intelligent tutoring systems: At the crossroads of artificial intelligence and education.* Norwood, N.J.: Ablex.

Gray, W.D. and Atwood, M.E. (in press). Transfer, adaptation and the use of intelligent tutoring technology: The case of Grace. In M. Farr & J. Psotka (eds.) *Intelligent computer tutors: Real world applications.* New York: Taylor & Francis.

Miller, J.R. (1988). The role of human-computer interaction in intelligent tutoring systems. In M. Polson & J. Richardson (eds.) *Intelligent tutoring systems.* Hillsdale, N.J.: Erlbaum.

Designing a Tool for Imaging Mental Models Underlying Training

Christopher J. Dede
Center for Interactive Educational Technology
George Mason University, Fairfax, Virginia 22030, USA

Abstract: The training process can be conceptualized as the student acquiring an evolutionary sequence of classification-problem-solving mental models. This paper describes the design themes underlying a computer-based tool, Mental Link, that would allow the evaluative imaging of such mental models. The fundamental design approach for this representational medium is traversal of virtual cognitive space: Typically intangible cognitive entities and links among them are visible as a three-dimensional web that represents a knowledge structure. Such a data-visualization approach has a high degree of flexibility and customizability to allow extension to other types of uses, such as a front-end for an intelligent tutoring system, knowledge base, hypermedia system, or semantic network.

Introduction: Mental Models in Training

Training centers on communicating a standardized package of knowledge and skills from an expert source to a learner. This process can be conceptualized as the student acquiring an evolutionary sequence of predetermined mental models of the subject domain. The type of mental model that underlies almost all training is classification problem solving (Clancey, 1984): Real world phenomena can be classified as entities in category systems nested hierarchically. Training a student to acquire this type of mental model involves communicating both the classification systems for a subject domain and the network of causal relationships among the entities in those categories.

Training is seldom depicted in this manner because educators do not have an adequate method for representing mental models. Typically, the core of training is portrayed by presenting outlines of subject headings or lists of goals and behavioral objectives. Unfortunately, these methods of summarizing focus primarily on individual, isolated pieces of the curriculum; they do not capture the nonlinear web of interrelationships among these integrated, hierarchically nested concepts and skills. Conceptualizing the training process as the communication of mental models requires synthesizing and depicting knowledge structures greater than the sum of their parts.

Discussing mental models is as fundamental to training as asking "what if" questions is to planning. A decade ago, the spreadsheet provided a medium for simulating financial alternatives, and business management strategies became much more effective. A similar software application is needed to empower the next generation of advances in training.

Imaging Mental Models

While adequate for presenting lists or outlines, sequential media such as text or speech are unsatisfactory when describing a mental model. Diagramming the interrelationships of multiple curricular components typically produces an indigestible tangle of visual spaghetti. Since causal links among entities across hierarchies produce very complex networked structures, a nonlinear medium with support for visualization techniques, navigation in virtual space, and imaging from multiple perspectives is required to represent mental models.

This paper describes the design themes underlying a computer-based tool, Mental Link, that images classification-problem-solving mental models. Hypermedia—a knowledge representation emerging from advances in information technology, artificial intelligence, and cognitive science—is used as the nonlinear representational architecture. Strategies from the field of scientific visualization have been applied to allow the presentation of complicated information without overwhelming the viewer. To empower intelligent responses from the tool, artificial intelligence capabilities, such as inferencing over the network structure, have been incorporated into the design.

The fundamental design approach underlying this mental modeling tool is traversal of virtual cognitive space. Typically intangible cognitive entities and links among them are visible as a three-dimensional web that represents a knowledge structure. The user can navigate through this virtual environment and also can transcend the metaphor of physical space by shifting among alternate contexts (e.g., informational, relational, diagnostic) that provide different perspectives on a particular cognitive entity. In addition, special capabilities to minimize complexity (i.e., guided tours, filters, hypertext, visualization features) are available.

Through embedded inference across the semantic networks in a mental model, Mental Link provides both a medium for communication and an empowering environment for individual and group work. Authoring functionalities are designed to make creating or extending a model simple, as the application generates virtual semantic structures with minimal input. The tool has a high degree of flexibility and customizability to allow extension to other types of uses, such as a front-end for an intelligent tutoring system, knowledge base, hypermedia system, or semantic network.

Design Principles

Below are summarized the design assumptions and principles underlying this prototype for a mental modeling tool. Many of these could generalize to any application involving traversal of virtual cognitive space.

Hypermedia as a Representation for Semantic Networks

The representational architecture selected for the mental modeler is a semantic network instantiated in a hypermedia system. This tool required a representational architecture that satisfied several constraints. First, its knowledge structure needed to mirror the associational webs underlying human long-term memory. Second, the formalism chosen had to be capable of representing a classification-problem-solving perspective on mental models. Third, the architecture needed to support an embedded inference engine to provide intelligent functionalities for users. Finally, the representational system had to facilitate revealing and concealing the complexity intrinsic in internested hierarchical structures.

A hypermedia system for depicting semantic networks best satisfied all these constraints. A semantic network is a representational architecture that provides a semi-structured mechanism for displaying associative interconnections (Woods, 1985). In hypermedia, nodes of information are interconnected in a nonlinear manner by links. This provides a framework for nonlinear representations of knowledge, as well as creating the potential of embedded inference across the network of interconnections (Halasz, 1988).

Such an architecture can represent the internested hierarchies typical of the classification-problem-solving perspective on training. In addition, hypermedia allows the user substantial control over the complexity and depth of the material being presented. Through activating a browsing capability, the user can access a global view of the hypermedia network; this simplifies traversing a complex cognitive map. Also, by activating or suppressing the contents of nodes, users can selectively access detailed data while retaining an overall perspective on the entire knowledge structure.

Researchers in cognitive learning (Jonassen, 1986), instructional design (Russell, Moran, and Jordan, 1987), and educational applications of semantic networks (Denenberg, 1988) are finding hypermedia useful as a representational formalism. External to the training community, a hypermedia system has been developed that supports representing a mental model as a network of associations among the elements of a strategic plan (Carlson and Ram, 1990). The design of Mental Link extends these approaches by creating a nonlinear, semantic architecture for the three-dimensional virtual cognitive space the tool uses as its visual interface.

Visualization of Virtual Cognitive Space

This mental modeling tool immerses the user in a three-dimensional virtual environment in which cognitive entities and their interrelationships are presented as images. Data visualization transforms the symbolic into the geometric, making normally intangible entities visible; such an approach is useful in situations where the amount of data is large, communication among co-workers is vital to task performance, and interacting with the data to shape its presentation can aid in interpretation. As one example, the Information Visualizer experimental interface uses color and real-time, three-dimensional, interactive animation to create information objects (Robertson, Card, and Mackinlay, 1991).

An emerging concept in visualization, cyberspace, seems well suited to the types of evaluative imaging requisite for depicting mental models, but poses many research issues (Benedikt, 1991). These include the costs and benefits of reifying information, space-time axiomatics in artificial realities, magic versus logic as principles underlying user actions, the presentation of the self in a virtual context for group work, the meaning of travel and action when translated from a physical to a symbolic domain, coordinate systems for (un)real estate, the form and meaning (semiotics) of data objects, three-dimensional user interface design, visual languages, alternative spatiotemporal metaphors from physical reality, and the architecture of multi-dimensional data spaces.

Evaluative imaging in a cyberspace environment seems naturally suited to coalescing the complex data typical of a sophisticated training process. A three-dimensional virtual space with icons, shadings, and colors can used to depict clustered, nested hierarchies and their interrelationships.

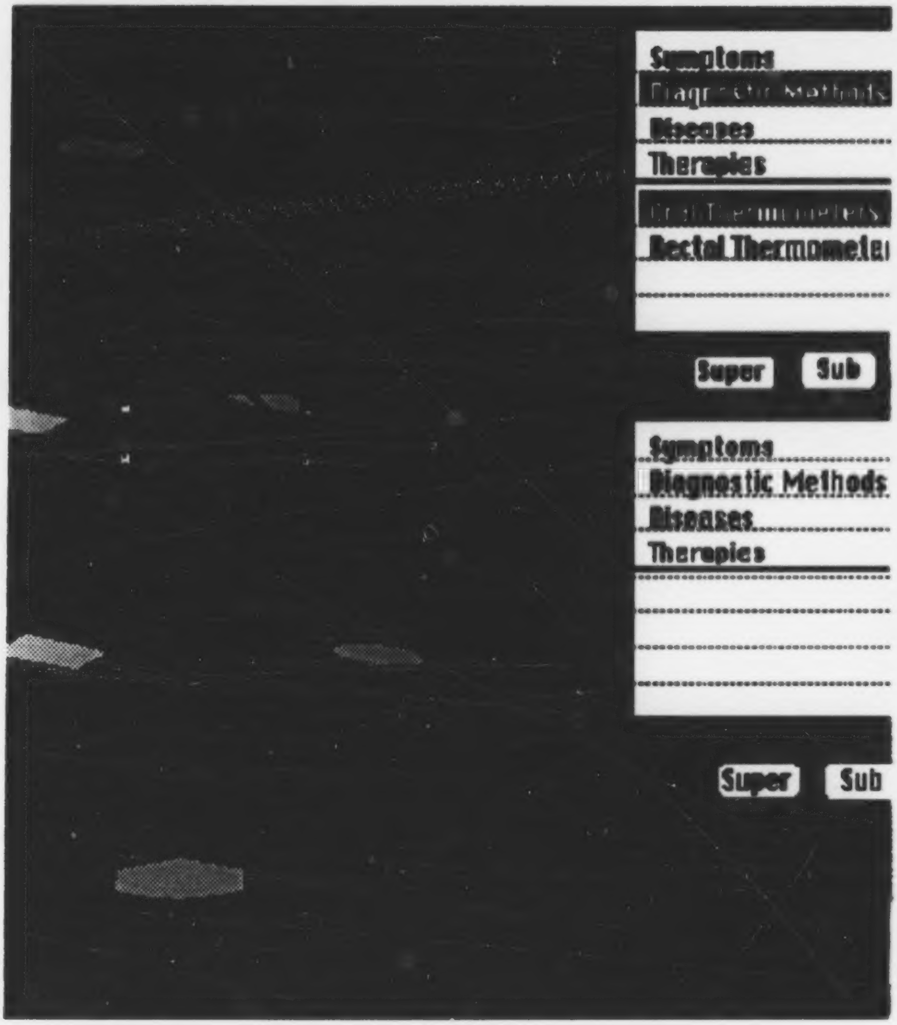

Symptoms
Diagnostic Methods
Diseases
Therapies
Oral Thermometers
Rectal Thermometer
Super
Sub
Symptoms
Diagnostic Methods
Diseases
Therapies
Super
Sub

Figure 1. Global view of medical model in Mental Link

Users can traverse and act on these cognitive structures through Mental Link's capabilities for semantic navigation.

Semantic Navigation

A major challenge for this mental modeling tool, as for any cyberspace application, is management of the computer-human interface. Complex configurations of multi-dimensional data must be converted to a spatiotemporal analog that allows users to transfer their usual strategies for navigation and action from physical contexts to cognitive space. User disorientation has been a particularly difficult problem in cyberspace work to date, because a nonlinear medium offers fewer clues to relative position and orientation than sequential forms of communication. While hypermedia representations offer some leverage by allowing user control over image complexity, large-scale semantic networks can present a cluttered and confusing appearance if viewed in full detail.

However, preliminary work in manipulating the visual presentation of large node/link structures indicates that interface functionalities can provide substantial aid in reducing the complexity of the image. For example, the SemNet tool developed at the Microelectronics and Computer Technology Corporation illustrates how controlled manipulations of associative webs can be used to reveal and conceal complexity by focusing on particular types of categories and relationships. This research is now generalizing into a theory of semantic navigation—based on multiple views of information and hierarchical object structures—that seems likely to reduce disorientation problems (Fairchild and Wexelblat, 1989).

In addition, the hypermedia community has been developing a variety of alternative strategies for facilitating user traversal through conceptual maps. Nielsen (1990) has constructed a system that uses two navigational dimensions, each with a different animation technique, to aid in orientation. The NoteCards group at Xerox's Palo Alto Research Center has been studying how visual cues embedded in the contents of hypertext nodes can create a guided tour through a complex web of information (Marshall and Irish, 1989). Brown University's Intermedia project has synthesized a variety of navigation approaches into a web-view capability for browsing (Utting and Yankelovitch, 1989).

Functionalities Incorporated into the Design

The design of Mental Link incorporates and extends many of the design strategies discussed above. By viewing hypermedia-based visual representations from varied perspectives and at different levels of granularity, the user can focus attention only on those aspects relevant to the current instructional issue. The multiple hierarchical structures characteristic of classification problem solving are mapped into spatial analogues whose physical characteristics aid orientation and navigation.

While the imaging of three-dimensional views of the entire cognitive space still presents many challenges in reducing complexity, the semantic navigation strategies in this mental modeling tool offer simple functionalities that empower sophisticated traversal and action.

In particular, Mental Link would support search, query, and inference over the entire knowledge base embedded in the network, including the descriptive information stored for each node and link in the structure. The tool could aid an author in maintaining the completeness and consistency of the knowledge network, as well as in creating concept maps that provide multiple perspectives on semantic structures. Further, embedded inference over the network allows customizing and tailoring the tool to specific purposes, such as making judgements about learning style based on an evolutionary sequence of mental models. Such intelligent aids are vital for preventing users from being overwhelmed by the scale, detail, and complexity inherent in the classification-problem-solving models that Mental Link images.

Limits on space preclude providing any detail on the functional specifications of Mental Link. The design details are described fully in Dede and Jayaram (1990), with its accompanying research videos and demonstration software. These materials simulate many of Mental Link's functionalities, so the user can experience a guided tour through the interface and most features of this potential application. However, these prototypes are unable to replicate the actual capabilities of the tool; they are merely demonstrations and

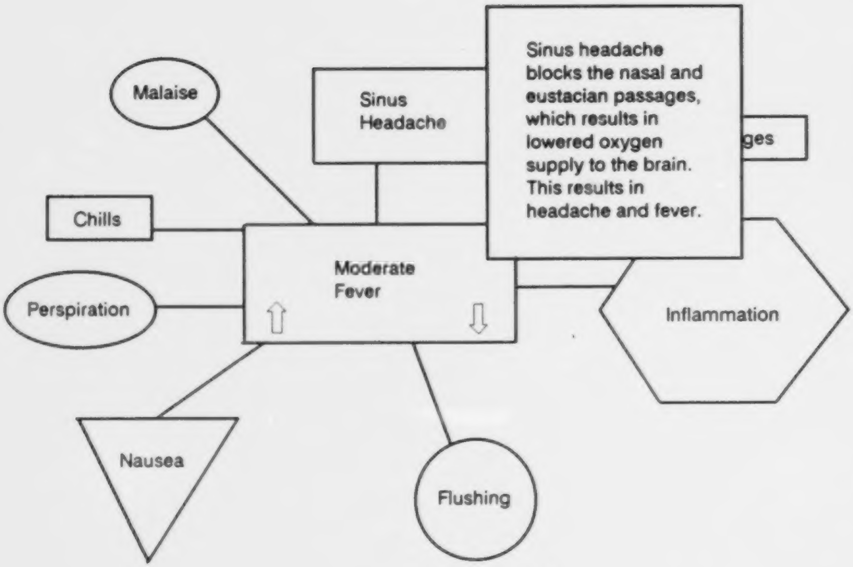

Malaise
Sinus Headache
Sinus headache blocks the nasal and eustacian passages, which results in lowered oxygen supply to the brain. This results in headache and fever.
ges
Chills
Moderate Fever
⇧
⇩
Perspiration
Inflammation
Nausea
Flushing

Figure 2. "Moderate Fever" entity in relational mode

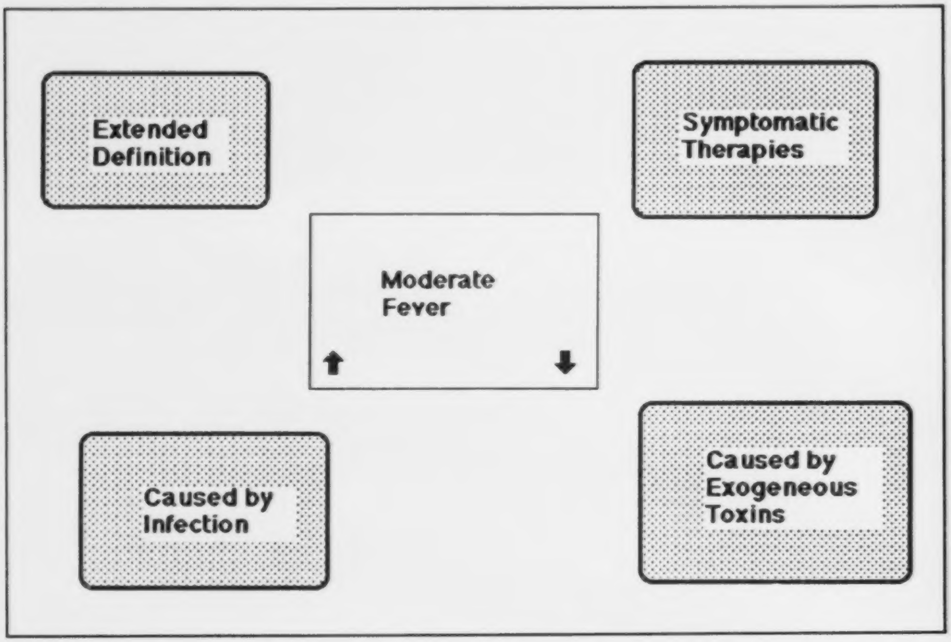

Extended Definition
Symptomatic Therapies
Moderate Fever
⬆
⬇
Caused by Infection
Caused by Exogeneous Toxins

Figure 3. "Moderate Fever" entity in informational mode contain no code that could be used in building Mental Link. This project is at the stage of presenting an image of what could be rather than supporting the construction of a working prototype.

Summary

Hypermedia representations of semantic networks, visualization of virtual cognitive space, and semantic navigation are the fundamental design principles underlying Mental Link. Combined, they provide a representational architecture for mental models different and more powerful than any medium now being used to conceptualize and evaluate the training process. These design strategies also suggest the foundations of a general theory for traversal of virtual cognitive space.

This research has been funded by the U.S. Air Force Human Resources Laboratory and the National Aeronautics and Space Administration Space Station Training Office. The views and conclusions contained in this report are those of the author and should not be interpreted as representative of the official policies, either express or implied, of NASA, the U.S. Air Force, or the United States Government.

Acknowledgements: Ms. Geetha Jayaram served as the Technical Documentation Specialist for this project and developed much of the demonstration software. At AFHRL, Lt. Colonel Hugh Burns, Major James Parlett, and Captain Mike Slaven were particularly supportive; at NASA-JSC, Barbara Pearson and Frank Hughes were very helpful as Technical Monitors.

References

Benedikt, M. (1991). Introduction. In M. Benedikt (Ed.), *Cyberspace: First steps.* Cambridge, MA: MIT Press.

Carlson, D.A. and Ram, S. (1990). HyperIntelligence: The next frontier. *Communications of the ACM* 33(3):311-321.

Clancey, W. (1984). Classification problem solving. In *Proceedings of the American Association of Artificial Intelligence.* Los Altos, CA: Morgan Kaufmann.

Dede, C.J. and Jayaram, G. (1990). *Designing a training tool for imaging mental models* (AFHRL Technical Report CR-90-80). Brooks AFB, TX: Air Force Human Resources Laboratory.

Denenberg, S.E. (1988). Semantic network designs for courseware. In D.H. Jonassen (Ed.), *Instructional designs for microcomputer courseware.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Fairchild, K.F. and Wexelblat, A. (1989). *Navigation through cyberspace* (MCC Technical Report Number STP-161-89). Austin, TX: Microelectronics and Computer Technology Corporation.

Halasz, F.G. (1988). Reflections on Notecards: Seven issues for the next generation of hypermedia systems. *Communications of the ACM* 31(7):836-855.

Jonassen, D.H. (1986). Hypertext principles for text and courseware design. *Educational Psychologist* 21(4):269-292.

Marshall, C.C. and Irish, P.M. (1989). Guided tours and on-line presentations: How authors make hypertext intelligible for readers. In *Proceedings of Hypertext '89*. New York: Association for Computing Machinery.

Nielsen, J. (1990). The art of navigating through hypertext. *Communications of the ACM* 33(3):298-310.

Robertson, G., Card, S., and Mackinlay, J. (1991). The information visualizer. In *Proceedings of CHI 91*. New York: Association for Computing Machinery.

Russell, D.M., Moran, T.P., and Jordan, D.S. (1987). The instructional design environment. In J. Psotka, L.D. Massey, and S.A. Mutter (Eds.), *Intelligent tutoring systems: Lessons learned*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Utting, K. and Yankelovitch, N. (1989). Context and orientation in hypermedia networks. *ACM Transactions on Information Systems* 7(1):58-84.

Woods, W.A. (1985). What's in a link?: Foundations for semantic networks. In R.J. Brachman and H.J. Levesque (Eds.), *Readings in knowledge representation*. Los Altos, CA: Morgan Kaufmann.

Computational Media as a Foundation for New Learning Cultures

Andrea A. diSessa
Graduate School of Education
Division of Math, Science and Technology Education
University of California, Berkeley, CA 94720

Abstract: There has been a lot of discussion of computers revolutionizing learning. Models of computer use that supposedly will instigate this change range from intelligent tutors, to microworlds and simulations, to whiz-bang multi-media presentation systems. The premise of our current work is that deep changes to learning and education can come about through computers providing a new and dramatically extended infrastructure to knowledge representation as experienced by learners and mentors. Such a new medium should be the basis of a computational extended literacy. Fundamentally, this literacy needs to be two-way. Media that are presentation-only systems are simply inadequate.

There are three difficult hurdles a new medium must overcome. First, it must be comprehensible. Like current literacy, it makes no sense to think such a powerful tool will be instantly learned. But it must be approachable, and it must allow the development of deep competence with it. Second, such a medium must be fundamentally useful. If a medium does not support extensive use from a broad range of people with a broad range of interests, I suggest it will simply languish. Programming, as currently experienced, fails this test. Finally, the platform will need to be deeply generative of new modes of learning and even new cultures involving it. Technology that just does what prior technology did, but a little better, won't make important changes.

I will review features of our proposed new computational medium, Boxer, and discuss how we think we are doing in the three challenging areas, discussed above. I will use as examples courses we or others have developed using Boxer. We find children learning new things, in surprising ways, and doing so, along with their teachers, with a deeper sense of ownership of the process.

CBMIP -- A Case-Based Mathematics Instructional Planner

Zhihui Du and Gordon McCalla
ARIES Laboratory
Department of Computational Science
University of Saskatchewan
Saskatoon, Saskatchewan S7N 0W0 CANADA

Abstract: A case-based instructional planner is described that produces teaching plans for the domain of ordinary differential equations. These plans can be used either by a human mathematics teacher or by an intelligent tutoring system. Experiments with the system show case-based planning to be a natural approach to instructional planning. The paper concludes that instructional domains present new challenges to case-based reasoning, but also provide new opportunities, too.

Introduction

People tend to recall and use past experiences to guide their problem solving in new situations. This observation has led to a number of research projects whose goal is to investigate how planning can be done through adaptation of old plans rather than via the more traditional STRIPS-style "first principles" planning through subgoal elaboration. Work by McCalla, Reid and Schneider (1982) on "plan splicing," and by Schank (1982), Hammond (1989) and others on "dynamic memory" and "case-based planning" have explored methodologies for doing this plan adaptation.

In our work we are investigating how case-based planning can be adapted to the needs of instruction. Based on Hammond's (1989) case-based planning architecture, we have developed a case-based instructional planner (CBMIP) to generate instructional plans for large scale courses in the domain of ordinary differential equations. These plans are at the content level (Wasson 1990), rather than the delivery level, in that they describe sequences of teaching actions to help the student learn concepts in the domain, but make no commitment as to how these teaching actions will actually achieve their instructional objectives in real interactions with the student. The plans could be used either by an intelligent tutoring system, or by a human teacher as a curriculum.

The Structure of CBMIP

The structure of the case-based instructional planner is given in Figure 1. The selector is responsible for retrieving a plan from plan memory that is relevant to the current instructional objectives. Instructional objectives are described in terms of five different characteristics of differential equations and their solution: equation type being solved, solution range, solution type required, coefficient of equation and variable type of the equation. These aspects are used as an index in order to find a known instructional goal that is the "best match" to the current objectives. Associated with this best matching goal is a

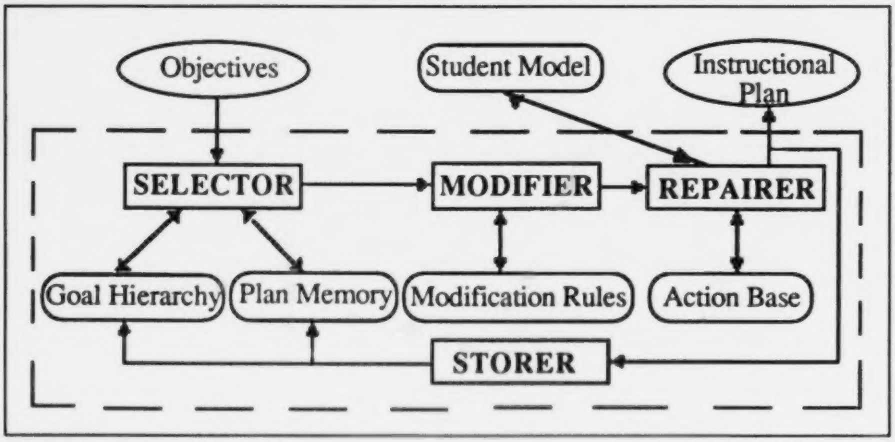


Figure 1. The structure of CBMIP

plan relevant to achieving the goal. This plan will form the first approximation of a plan to meet the current instructional goal, but may have to be modified because of differences between the current goal and the goal for which the plan was originally developed.

This modification is carried out by the modifier. Each difference is reduced using modification rules and modification actions that change the plan appropriately. The modifier reduces these differences independently of each other and pays little attention to conflicts among subplans that might arise as a result of mutually interacting modifications.

The repairer is responsible for eliminating these conflicts and filling in any gaps that arise in the plan. Repair is achieved in two stages: resolving conflicts within the plan and making changes according to a student model. If an action is found to be redundant or irrelevant to the current instructional goal, then the repairer removes that action according to the repair strategy. If the action is in the wrong position, then the repairer finds the correct position and puts it there. In some cases the repairer will add appropriate actions into the plan to erase conflicts.

Once the plan is fully developed, it is output to the user or to the application system. It is also passed on to the storer to be saved for future use. The storer doesn't simply store the plan, but also must appropriately index it so that it can be retrieved, modified, and repaired in the future just like a plan that has been built-in to the system from scratch. Thus, the storer is actually a learning component, in that it collects and stores the knowledge that the planner has learned during the planning process. The storer will not be discussed further in this paper; details can be found in Du (1991).

Knowledge Representation

There are a number of knowledge structures used in the system. The instructional plan produced by CBMIP is expressed in terms of teaching actions. Each teaching action has a name, prerequisite knowledge required by students before this action is relevant, and post-requisite knowledge that can be assumed will be known by students after the action is executed.One such action is **Taction35**:

Taction35
Action-Name: Finite Base Function Set (FBFS)
Precondition: Independent Set (ISET), Orthogonality (ORTH)
Effect: Finite Base Function Set (FBFS)

The core of each plan is a sequence of teaching actions which can be represented in a graph with sequencing links (see Figure 2). Each plan also has a name, an indication of the goal it can directly achieve, and two sets: a set of characteristics for which it is definitely relevant and a set of characteristics for which it is definitely not relevant. For example, **Plan4** looks like

Plan4
Plan-Name: Instruct Basic Finite Element Analysis
Goal-Achieve: Goal33
Rel-Characteristics: approximation, (global infinite),
(special general), (constant variable exp poly trig),
(single multiple)
Not-Rel-Characteristics: exact, local
Tactions: <the graph in Figure 2>

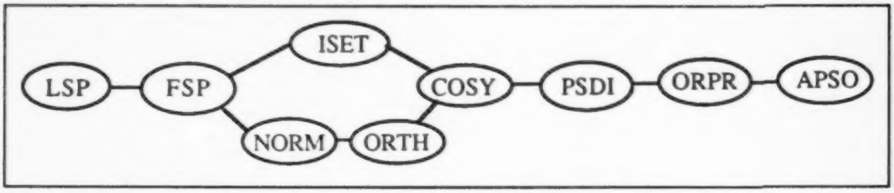


Figure 2. The graphical representation of an instructional plan

The goals are hierarchically organized into a discrimination network (see Figure 3). The various characteristics of differential equations.label discrimination arcs at the upper levels of the hierarchy. Specific instructional goals form terminal nodes in the hierarchy. Associated with each specific instructional goal is its name, characteristics, and the name of a plan that supports achieving this goal. This plan may not achieve the goal directly, but will be relevant to achieving the goal. One such goal is **Goal33**:

Goal33
Goal-Name: Basic Finite Element Approximation
Solution-Type: approximation
Solution-Range: global
Equation-Type: special
Variable-Type: single
Coefficient-Type: constant
Support-Plans: Plan4

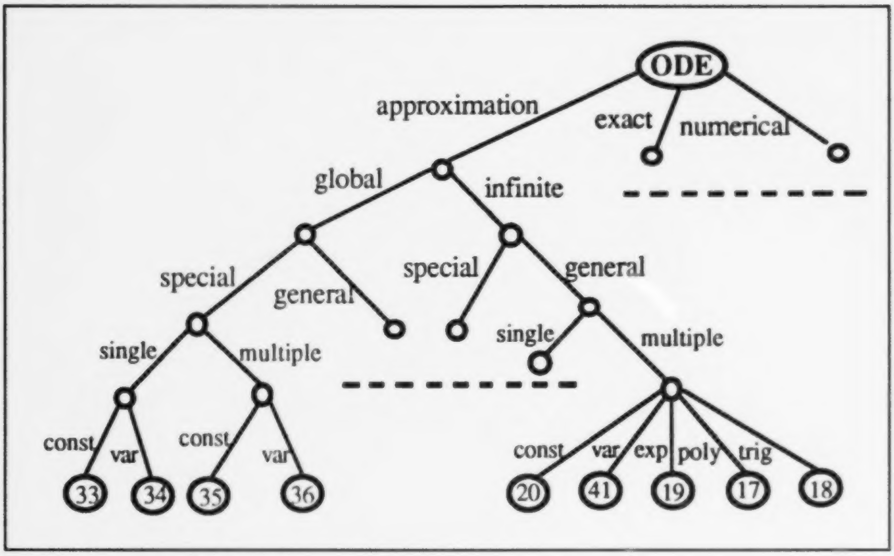

ODE
approximation
exact
numerical
global
infinite
special
general
special
general
single
multiple
single
multiple
const
var
const
var
const
var
exp
poly
trig
33
34
35
36
20
41
19
17
18

Figure 3. The goal hierarchy in CBMIP

The modification rules are a set production rules that are used to reduce differences between plans. Each modification rule has a name, a description of the differences this rule will reduce, and a set of modification actions that actually change the plan. For example, **Mrule6** looks like

Mrule6
Name: Extend Constant To Variable
Goal-Diff: (constant variable)
Mactions: Maction-7, Maction-8, Maction-11

Associated with each modification action is a label indicating the characteristic most relevant to this Maction; this is used as a heuristic for choosing among various actions in a particular situation. An Maction also has an old environment slot, indicating teaching actions that should be deleted from the instructional plan, and a new situation slot indicating teaching actions that are needed in the current situation.. For example **Maction8** looks like

Maction8
Label: approximation
Old-Env: Coordinate Systems
New-Situ: Finite Base Function Set

A Planning Example

Suppose CBMIP is given objectives to teach about approximation solutions of multi-variable differential equations in an infinite range, i.e., the characteristics are (**approximation, infinite, general, multiple, variable**). The selector searches the goal hierarchy and returns **Goal41** as a goal that will meet these objectives. (If the objectives had been broader, there could have been many relevant goals; one would be chosen.)

Goal41
Goal-Name: General Infinite Multiple Approximation
Solution-Type: approximation
Solution-Range: infinite
Equation-Type: general
Variable-Type: multiple
Coefficient-Type: variable
Support-Plans: Plan4

The achievement of **Goal41** is supported by **Plan4**, but according to its Goal-Achieve slot. **Plan4** can directly achieve only **Goal33**. **Goal33** and **Goal41** have differences in solution range, equation type,variable type and coefficient type as summarized in this list:

((global infinite) (special general)
(single multiple) (constant variable))

These differences must be eliminated if **Plan4** is to be used to achieve **Goal41**.

The modifier is used to erase these four differences. The modification rule base is searched to find an appropriate Mrule to erase the differences one by one until all differences are eliminated. For example, to erase the difference (**constant variable**), **Mrule6** is called, because the Goal-Diff of **Mrule6** is (**constant variable**). Once an Mrule is found, the corresponding modification actions are executed to actually modify the selected plan. The modifier tries to choose the best Maction using heuristic knowledge in the label slot. If the label slot contains characteristics relevant to the current instructional goal, then this Maction is considered as the best Maction. For example, there are 3 Mactions (**Maction7 Maction8 Maction11**) associated with **Mrule6**, but the label slot of **Maction8** is "approximation" which is one of the characteristics of the current instructional goal, so it is considered to be the best one.

Maction8 consists of two phases: one to insert node FBFS into the selected plan, the other to delete node COSY from the selected plan. After executing Maction8, an intermediate modified plan is created (see Figure 4 (i)). There is a gap between node FBFS and node ORPR (we use a dashed line to represent the gap between nodes in Figure 4 (i)). The occurrence of gaps reflects the fact that the addition and deletion of teaching actions from a plan will often result in inappropriately matched effects of one action and preconditions of the next action(s). Any gap within the plan will be filled by the repairer later.

There are still three differences left to resolve:

((global infinite) (special general) (single multiple))

These differences can be similarly reduced using other Mrules and Mactions, resulting in a final modified plan shown in Figure 4 (ii). Note that in the plan in Figure 4 (ii) the gap between FBFS and PSDI no longer exists, since sebsequent modification actions have deleted ORFS after FBFS. This points out that delaying repair until all modification actions have been undertaken is worthwhile since repairs can be made irrelevant by subsequent modification actions.

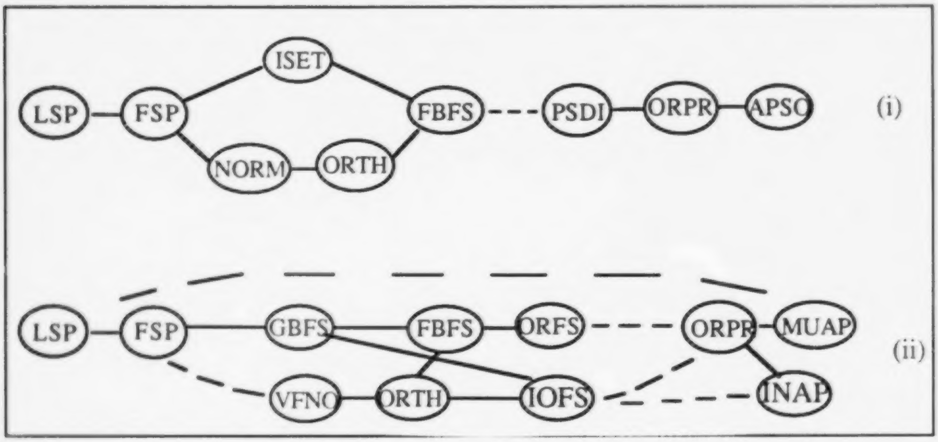

ISET
LSP
FSP
FBFS
PSDI
ORPR
APSO
(i)
NORM
ORTH
LSP
FSP
GBFS
FBFS
ORFS
ORPR
MUAP
(ii)
VFNO
ORTH
IOFS
INAP

Figure 4. The intermediate and final modified plan

The fully modified plan is sent to the repairer. The repairer attempts to fill each gap by finding an instructional action (or a chain of instructional actions) whose precondition matches the effect of the node at the begining of the gap and whose effect matches the precondition of the node at the end of the gap. For example, there is a gap between node FSP and VFNO. Because the precondition of action ABPM matches the effect of node FSP and the effect of action ABMP matches part of the precondition of node VFNO, action ABPM is inserted into the plan to fill the gap. The other part of the precondition for node VFNO can be matched by node MULC, so MULC is added to the plan to fully specify the required preconditions of node VFNO (see Figure 5 (i)). As a by-product of this activity the gap between LSP and MUAP is also closed because ABPM has other effects that meet the required preconditions of MUAP. After all gaps are similarly filled, the final plan in Figure 5 (ii) results.

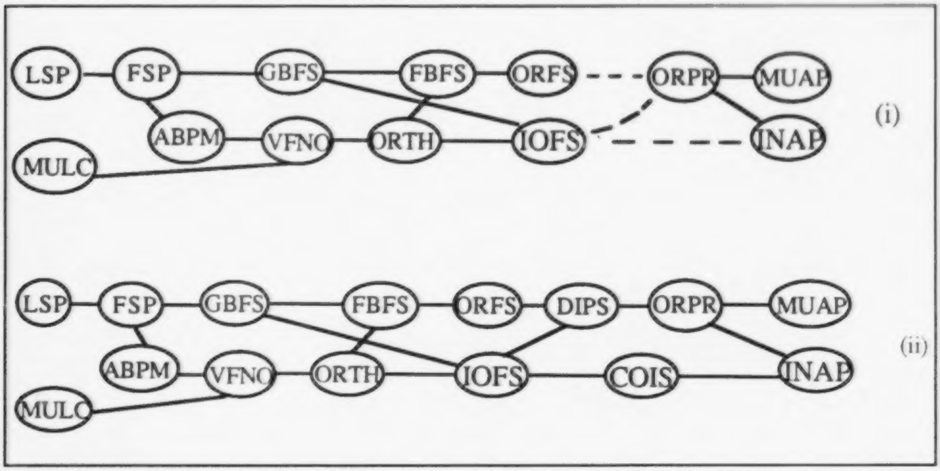

LSP
FSP
GBFS
FBFS
ORFS
ORPR
MUAP
(i)
ABPM
VFNO
ORTH
IOFS
INAP
MULC
LSP
FSP
GBFS
FBFS
ORFS
DIPS
ORPR
MUAP
(ii)
ABPM
VFNO
ORTH
IOFS
COIS
INAP
MULC

Figure 5. The intermediate repaired plan and final repaired plan

While the plan at this stage will completely achieve the instructional goal according to relationships known to exist within the domain, it has not been adjusted to take into account particular idiosyncracies of individual students. If it is desired to individualize the instruction for a particular student, a further round of repairs can be undertaken using a student model to determine extra actions to incorporate into the plan or redundant actions to eliminate from the plan. Leaving the individualization aspects until the final phase allows a plan to be produced without individualization if a human teacher is to use it as a curriculum, and with individualization if it is to be used by an intelligent tutoring system.

After an instructional plan is successfully created, the storer is called to store the plan into plan memory. Goals and plans are also updated, so that if the same goal is met later the successful plan can be directly selected and be used to achieve the instructional goal.

Conclusions

The system has been implemented in IBUKI Common Lisp, and in its current version incorporates 44 instructional goals, 64 instructional actions, 4 plans, 9 Mrules and 13 Mactions to handle the domain of ordinary differential equations and related concepts. Currently, some 40 different plans have been produced to achieve various instructional objectives.

This research demonstrates how case-based reasoning can be applied to intelligent tutoring. Case-based planning is a methodology that seems natural for instructional domains (in contrast to "first-principles" instructional planning such as that carried out by Peachey and McCalla (1986)). Moreover, the case-based system, through its ability to store away plans, can in some sense learn. The system can be used in at least two ways: to produce instructional plans for human teachers to use, or to produce instructional plans to be used by an intelligent tutoring system. This has necessitated a two-phase repair process that doesn't occur in other case-based approaches: the first phase produces the non-individualized plan useful for the human teacher, while the second phase individualizes this plan for the one-on-one tutor. The distinction between specific instructional goal (e.g., teach general infinite multiple approximation) and coarser-grained instructional objectives (e.g., teach something about approximation solutions in an infinite range) is also interesting in that it allows the CBMIP user to focus on the skills the student is to learn, not the specific content that exercises these skills.

Our experiments so far have shown the utility of case-based reasoning to a mathematics domain, but further experiments are required to fully explore all of the ramifications of this approach in this domain. Beyond this it is important to see if the approach can be extended to other domains. We are optimistic that this will not prove to be too difficult.

Acknowledgements: We would like to thank the Natural Sciences and Engineering Research Council of Canada for funding this research and the Canadian International Development Agency for funding the first author's graduate studies.

References

Du, Z. (1991). *Case-based course planning: An experiment in producing instructional plans in a mathematics domain.* Saskatoon, Canada: M.Sc. Thesis, Department of Computational Science, University of Saskatchewan.

Hammond, K.J. (1989). *Case-based planning*. San Diego: Academic Press.

McCalla, G.I., Reid, L. and Schneider, P.F. (1982). Plan creation, plan execution and knowledge acquisition in a dynamic microworld. *Int. J. of Man-Machine Studies* 16: 89-112.

Peachey, D.R. and McCalla, G.I. (1986). Using planning techniques in intelligent tutoring systems. *Int. J. of Man-Machine Studies* 24 (1): 77-98.

Schank, R.C. (1982). *Dynamic memory : A theory of learning in computers and people*. Cambridge, England: Cambridge University Press.

Wasson (Brecht), B.J. (1990). *Determining the focus of instruction: Content planning for intelligent tutoring systems*. Saskatoon, Canada: Ph.D. Thesis, Department of Computational Science, University of Saskatchewan.

The Space Shuttle Fuel Cell Tutor: A Hybrid Training Environment

Phillip C. Duncan, Ruston M. Hunt, Robert J. Glushko, and William B. Rouse
Search Technology, Inc.
4725 Peachtree Corners Circle, Norcross, GA 30092

Abstract: A technical training environment has been developed and fielded at the NASA Johnson Space Center. This environment, known as the Space Shuttle Fuel Cell Tutor integrates several technologies, a mixed-fidelity simulation, cooperative expert systems, and a hypermedia interface. The Space Shuttle fuel cell was selected as the technical domain for three reasons: (1) the dynamics of this complex system are often confusing for astronauts to learn; (2) astronauts do not have the luxury of shutting the system down, often they must operate, troubleshoot, and repair the system simultaneously; and (3) traditional training which emphasized procedural knowledge was found to be brittle or incomplete in novel situations. The Fuel Cell Tutor addresses the problem of complexity by representing a professional trainer's mental model of the system. This pictorial model allows for progressive disclosure of information and serves as the entry point for additional technical information. The problem of operating and diagnosing system problems simultaneously necessitated the creation of two cooperating expert systems–an operator and a troubleshooter. Finally, to overcome the problem of brittle knowledge, a dynamic simulation provides information about the deep structure and function of the system. All of this information is accessible through the hypermedia interface.

Introduction

The cockpit of the NASA Space Shuttle is one of the most complex man-machine operating environments in the world. The Space Shuttle astronauts not only have to fly the vehicle, but also need to know how to troubleshoot problems and fix them. The reason, of course, is that the Space Shuttle astronauts cannot conveniently land and have a mechanic fix the problem for them. The duration of flight for the Space Shuttle is much longer than that for most aerospace craft, consequently there is more time for things to go wrong. Another unique aspect of the Space Shuttle is that the power supply, comprised of three fuel cells, is very different from traditional batteries and electrical generators. Despite the unusual nature of the Space Shuttle, the requirement for astronaut familiarity with the operational and functional aspects of the vehicle is unmatched by military or civilian pilots.

The Flight Training Branch of NASA's Johnson Space Center selected the Space Shuttle fuel cell as representing its most difficult training problem for astronauts. Not only is the fuel cell unlike other power supplies in function, but it is operationally demanding and absolutely essential for crew survival and mission success. The chemical reaction which the fuel cell uses to produce electricity and water is delicate. The temperature of the stacks where the catalytic reaction between hydrogen and oxygen takes place must be kept within a narrow range. If the stack becomes too cold, the thin walls of the cell could freeze and crack, destroying the cell. If the stack becomes too hot, the reactants could leak and potentially explode. If the stack is compromised toxic chemicals can enter the astronauts drinking water supply. Power means life aboard the Space Shuttle and if even one fuel cell malfunctions that means a loss of power to computers and other critical equipment. Several launches have been delayed and one mission returned prematurely due to failure of only one of the three fuel cells. The most notable recent example of fuel cell malfunction was the delay which almost grounded the Discovery from its historic return to flight.

Reducing Complexity with Appropriate Mental Models

There are two characteristics of complexity which make learning a system difficult for NASA's astronauts. The first is the large number of system components and the second is the degree of interaction among the components. We first discuss simplifying representations of the system components and then we discuss the simulation.

Reducing the number of components is achieved by appropriate use of abstraction and aggregation (Frey, Rouse & Garris, 1991). Astronaut trainers know which level of abstraction is the best for describing the system operation and which system components can be aggregated into function units. The process of capturing a trainer's mental model (or models) results in a representation which has multiple levels of abstraction and aggregation.

Kieras and Bovair (1984) have shown that learning the correct procedures for operating a novel device is facilitated by using an appropriate mental model. Gentner and Gentner (1983) have shown that learners draw different inferences based upon their mental model. As important as mental model research is for training, there are a number of limitations to that research because of the difficulty "capturing" mental models (Rouse & Morris, 1986), knowing what mental models should be taught (Kieras, 1988; Streitz, 1988) and how they should be taught (Eisenstadt, 1989; Moyse, 1989).

Although the process of capturing mental models is intricate, we have found that it is easier to capture progressively more complex models than one ultimate explanatory model. These models are then linked together from general to specific operational units (abstraction) and from few to many functional components (aggregation). Models are nested within models with the most detailed representation, a schematic diagram, at the remotest level. As the learner wants progressively more information more detail is disclosed until the learner is satisfied.

The interaction of system components is the second characteristic of complexity. While there are numerous failures which may be simulated in the Fuel Cell tutor, the potential interactions have been reduced by simulating only one failure per learning scenario. The learner encounters combination malfunctions in other training simulators.

The Fuel Cell tutor simulator contains all of the computer displays, gauges, and panels which astronauts see in the Space Shuttle. The needles on the gauges fluctuate, the data on the CRTs change in real time, all of which are designed to portray a realistic picture of the fuel cell system. The Fuel Cell tutor simulator uses fairly standard technology and will not be further elaborated here.

Before the Fuel Cell tutor, two other intelligent systems for technical training, STEAMER and QUEST, have based their approach on the explicit use of mental models and simulations. STEAMER (Williams, Hollan & Stevens, 1981) allows the learner to inspect various layers of a functional model and primarily focused on the graphical interface. QUEST (White & Frederiksen, 1984, 1985) utilizes a succession of mental models to gradually guide the learner through increasing levels of functional complexity. However, neither of these systems have attempted to integrate procedural and functional mental models.

Learning One System from Two Experts

The knowledge required to operate a complex system is different from the knowledge required to diagnose and repair malfunctions in the system. The Space Shuttle fuel cell is no exception. A familiar example of the difference between operating and maintaining knowledge, is the difference between knowing how to drive a car versus knowing how to repair a car. Car drivers abound, but a mechanic who can diagnose and repair car problems is a precious commodity.

Norman (1983) has discussed the different mental models held by operators and maintainers. Operators are more concerned with the procedural aspects of a system while maintainers are more concerned with the functional aspects. It is our hypothesis that experts have an integrated mental model including both procedural and functional aspects. Further we assert that if experts have an integrated model there exists an efficient means for novices to acquire that expert model. This hypothesis is the theoretical underpinning of the Fuel Cell tutor research.

To address this problem, the Fuel Cell tutor (Johnson, Norton & Duncan, 1988) integrates procedural and functional mental models and provides the learner with multiple viewpoints. A viewpoint is defined as an interpretive context (Wenger, 1987, p. 355) which the learner uses to develop a mental model. In this case of the Space Shuttle fuel cell we want to provide the learner two viewpoints, a procedural viewpoint and a functional viewpoint.

The importance of training with multiple viewpoints was seen by NASA's flight training branch. One of the problems they encountered training astronauts procedures without the accompanying functional knowledge, is that the astronauts were only able to respond effectively to contingencies which had been predicted and described. Worse still, without functional knowledge the astronauts would have no basis for even attempting to solve unexpected problems involving the fuel cell.

The Fuel Cell tutor has two expert systems which are very different in structure. The functional expert system employs a matrix-based inference engine which relies on the functional connectivity of the components to determine the best troubleshooting advice. The procedural expert is a conventional production rule-based expert system. Yet, the two experts can communicate with each other due to their common computer language. This language is C Language Integrated Production System (CLIPS) which was developed by NASA to allow rule-based systems to be integrated with sequential C programs.

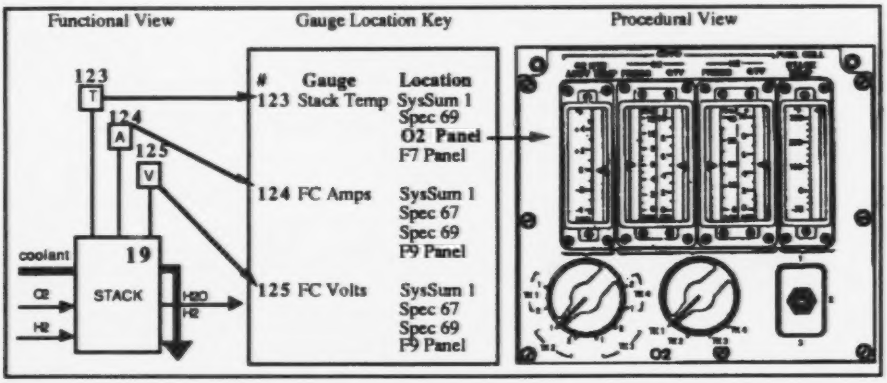
The basic unit of communication common to both experts is the "observation." All of the information observable from the 13 fuel cell gauges, panels, and displays have been broken down into discrete units. For example, on a computer display called System Summary 1, there is information pertaining to the stack temperature. This constitutes the unit of information we refer to as an observation.

As the learner looks at a display or gauge for the first time a row of X's conceal the underlying information. The student selects the area marked by the X's to reveal the information. In this manner, the Fuel Cell tutor verifies that the learner has observed the information. Once the learner has observed the information, messages are passed updating the procedural and functional experts. The procedural expert can then generate specific advice which will take observed information into account and not suggest procedures which would be redundant for the learner. The functional expert updates the components which the learner has seen and suggests applicable troubleshooting advice. By constantly updating the experts with learner observations the Fuel Cell tutor can respond with intelligent advice.

Linking it All Together with a Hypermedia Interface

The Fuel Cell tutor has two representations of the procedural viewpoint. A high-level view depicts the relative location of the displays and gauges in the Space Shuttle cockpit and another view is a close-up of each. The functional viewpoint is also represented in multiple ways. The Fuel Cell tutor contains a schematic of the fuel cell system, derived from a NASA training manual. The schematic representation has pop-up windows describing each component. A higher level view indicates functionally related components and component modules as well as a reference to physical display and gauges.

Transfer between viewpoints is facilitated by a common representation. The common representation in the Fuel Cell tutor is a diagram which indicates the functional relationships between components and the corresponding gauge (or display) locations. See Figure 1.



#	Gauge	Location
123	Stack Temp	SysSum 1 Spec 69 O2 Panel F7 Panel
124	FC Amps	SysSum 1 Spec 67 Spec 69 F9 Panel
125	FC Volts	SysSum 1 Spec 67 Spec 69 F9 Panel

Figure 1. Composite diagram indicating the sources of transfer between function and procedural viewpoints

In addition to the common representation of the Fuel Cell tutor there is another means whereby transfer occurs. The learner receives consistent advice from the two experts, the functional expert and the procedural expert. The earlier example of the malfunctioning stack temperature sensor shows that the experts can be queried sequentially. Learners may also ask the two experts for advice at the same point in the diagnosis for comparison.

An Actual Example

We have already mentioned the dire effects of an aberrant stack temperature. Let's use that as an example for a comparison of the two viewpoints of the Fuel Cell tutor. The learner begins a training scenario in the Fuel Cell tutor and immediately an alarm sounds which indicates that the problem is serious, endangering the Space Shuttle's mission or the crew itself. The learner asks the procedural expert for advice. The expert replies that the best initial procedure is to glance at the annunciator panel (F7) front and center on the flight deck. Further the expert suggests that this procedure is generally effective for narrowing the malfunction down to a specific subsystem. This meta-level knowledge is more appropriate for novice astronauts than experts, because experts would almost instinctively glance at the annunciator panel. But it reinforces the behavior and provides a rationale for the procedure instead of requiring rote memorization of procedures.

The learner glances at the annunciator panel and notices that the STACK TEMP light is indicated. Now suppose the learner wants to explore the deeper functional level of the fuel cell stack. At this point a command will take the learner to a schematic of the fuel cell system where information about the functions of components is available. Now the learner may ask the functional expert for advice about what is most likely responsible for the malfunction. The functional expert guides the learner's troubleshooting, providing clues about the feasible set of components responsible for the malfunction, and providing an exploratory laboratory for testing those components. In this particular example there are a limited number of components which could fail that would cause the stack temperature to fluctuate dangerously. Figure 1 indicates the components (11-21) which are functionally related to stack temperature. The prominent feature of Figure 2 is the coolant loop.

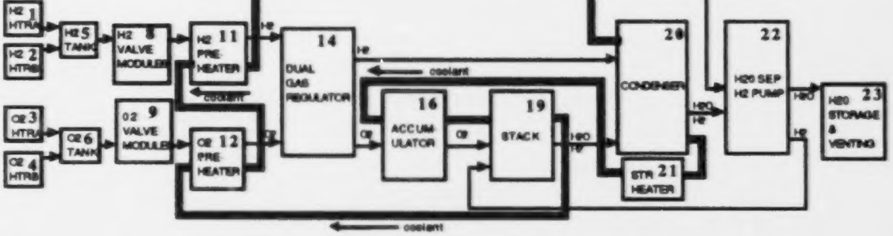

MN
AC
10 RELAY
ac power
ac power
ECU 7
13 COOLANT PUMP
15 FREON LOOP
17 FREON LOOP
18 COOLANT PREMIX VALVES
coolant
coolant
H2 1 HTR A
H2 2 HTR B
H2 5 TANK
H2 8 VALVE MODULE
H2 11 PRE-HEATER
H2
14 DUAL GAS REGULATOR
H2
coolant
coolant
O2 3 HTR A
O2 4 HTR B
O2 6 TANK
O2 9 VALVE MODULE
O2 12 PRE-HEATER
O2
O2
16 ACCUMULATOR
O2
19 STACK
H2O H2
20 CONDENSER
H2O H2
21 STR HEATER
22 H2O SEP H2 PUMP
H2O
H2
23 H2O STORAGE & VENTING
coolant

Figure 2. Simplified diagram of components affecting stack temperature (gauge locations excluded)

Next the learner may ask for advice about the function of the system. The functional expert knows that the learner has seen the STACK TEMP annunciator light and concludes that problem must be functionally upstream from the stack, or with the stack itself. The functional expert responds by suggesting the troubleshooting test which will eliminate the most possibilities. In this case the functional expert suggests testing the input to the condenser. If condenser input is normal, then all of the components pertaining to the cryogenic reactants (1-6, 8-9, 11-12,14) must be normal. The learner makes the test and the cryogenic reactant subsystem is eliminated from consideration.

Having narrowed the problem down somewhat, the learner will want to determine the actual stack temperature reading in the coolant loop. (To emphasize the functional relationships, Figure 2 shows a diagram where the gauges have been removed.) The actual fuel cell diagram which NASA personnel use has the gauges indicated and the learner can see that the stack temp information is portrayed on two CRT readouts, System Summary 1 and Spec 69- Fuel Cell or on overhead panel O2. (See Figure 3.)

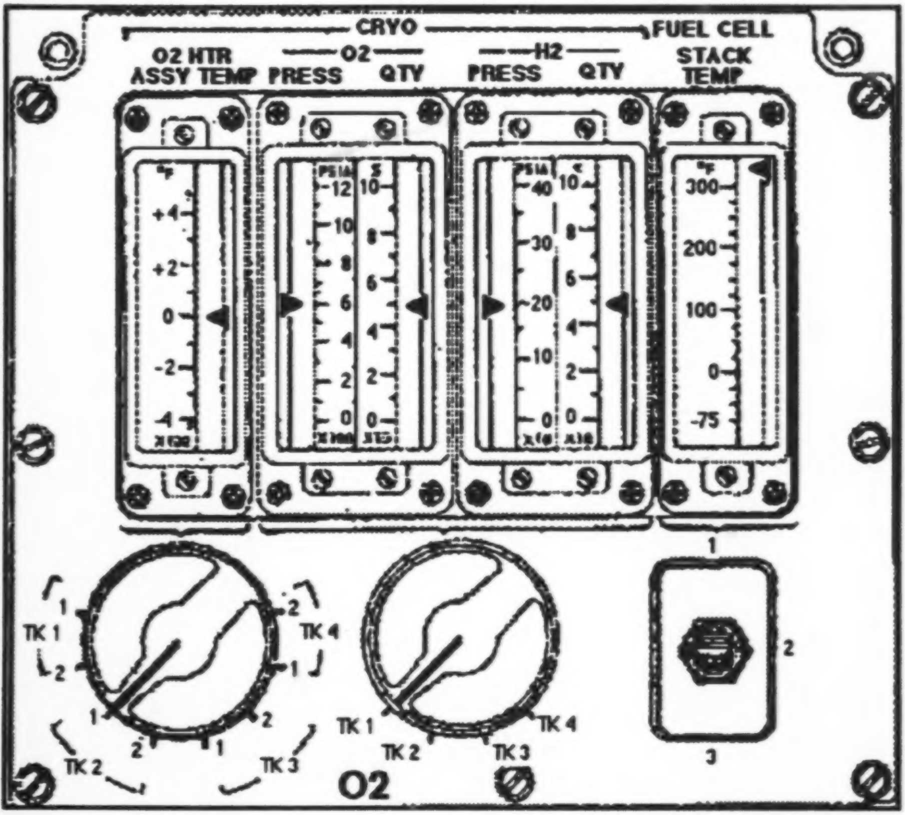

CRYO
FUEL CELL
O2 HTR ASSY TEMP
O2
PRESS
QTY
H2
PRESS
QTY
STACK TEMP
°F
+4
+2
0
-2
-4
X100
PSIA
12
10
8
6
4
2
0
X100
%
10
8
6
4
2
0
X10
PSIA
40
30
20
10
0
X10
%
10
8
6
4
2
0
X10
°F
300
200
100
0
-75
1
2
3
TK 1
TK 2
TK 3
TK 4
1
2
TK 1
TK 2
TK 3
TK 4
O2

Figure 3. Overhead panel 2 indicating off-scale high stack temperature

The learner may view either of the displays or the overhead panel which will indicate the stack temperature. The learner finds that the stack temperature is off-the-scale high! This is a situation where a less-experienced astronaut might begin to panic and shut down the fuel cell to prevent explosion. A more experienced astronaut would compare the CRT displays and the gauge reading to see if they correspond. In this particular training scenario, the CRT readings are in the normal range but the O2 gauge is off-the-scale. The expert astronaut correctly identifies the problem as a faulty temperature sensor.

Conclusion

The Fuel Cell Tutor is a practical example of a technical training environment which has successfully solved several training problems. It provides various representations of a complex system with different levels of abstraction and aggregation to reduce the complexity which the learner encounters at one time. A dynamic simulation further reduces potential complexity by displaying only one malfunction at a time. The Fuel Cell tutor combines functional and procedural viewpoints of the same system to enable the learner to reach a more complete understanding of the system. Each of these viewpoints is supported by its own expert system. The resulting hybrid training environment trains astronauts to combine procedural knowledge with knowledge about how the system functions to better prepare them for the uncertainty of space flight.

References

Eisenstadt, M. (1989). (AI in (Education in AI)): What does 'domain visualization' have to offer ITS? In D. Bierman, J. Breuker & J. Sandberg (Eds.) *Artificial Intelligence and Education: Proceedings of the 4th International Conference on AI and Education*, May 1989, Amsterdam, Netherlands, 100.

Frey, P.R., Rouse, W.B. & Garris, R.D. (1991). Big graphics and little screens: Designing graphical displays for maintenance tasks. To appear in *IEEE Transactions on Systems, Man, and Cybernetics*.

Gentner, D. & Gentner, D.R. (1983). Flowing waters or teeming crowds: Mental models of electricity, In D. Gentner & D.L. Stevens (Eds.) *Mental Models*. Lawrence Erlbaum Associates: Hillsdale, NJ.

Johnson, W.B., Norton, J.E., & Duncan, P.C. (1988). *An intelligent tutoring system for space shuttle diagnosis*. Conference on Space Operations Automation and Robotics, Dayton, OH: The United States Air Force and The National Aeronautics and Space Administration, July 20.

Johnson, W.B., Hunt, R.M., Norton, J.E., & Duncan, P.C. (1988). *Development and demonstration of microcomputer intelligence for technical training*. AFHRL Technical Report AFHRL-TP-88-8. Training Systems Division, Brooks AFB, TX.

Kieras, D.E. (1988). What mental model should be taught: Choosing instructional content for complex engineered systems. In J. Psotka, D. Massey & S.A. Mutter (Eds.) *Intelligent Tutoring Systems: Lessons Learned* (pp.85-111). Hillsdale, NJ: Lawrence Erlbaum Assoc.

Kieras, D.E. & Bovair, S. (1984). The role of mental models in learning to operate a device. *Cognitive Science*, 8, 255-273.

Moyse, R. (1989). Knowledge negotiation implies multiple viewpoints. In D. Bierman, J. Breuker & J. Sandberg (Eds.) *Artificial Intelligence and Education: Proceedings of the 4th International Conference on AI and Education*, May 1989, Amsterdam, Netherlands, 100.

Norman, D.A. (1983). Some observations on mental models. In D. Gentner & A.L. Stevens (Eds.), *Mental Models* (pp.7-14). Hillsdale, NJ: Lawrence Erlbaum Assoc.

Rouse, W.B. & Morris, N.M. (1986). On looking into the black box: Prospects and limits in the search for mental models. *Psychological Bulletin, 100* (3), 349-363.

Streitz, N.A. (1988). Mental models and metaphors: Implications for the design of adaptive user-system interfaces. In H. Mandl & A. Lesgold (Eds.) *Learning Issues for Intelligent Tutoring Systems*. Springer-Verlag: New York.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems: Computational and cognitive approaches to the communication of knowledge*. Los Altos, CA: Morgan Kaufmann.

White, B.Y. & Frederiksen, J.R. (1984). Modeling expertise in troubleshooting and reasoning about simple electrical circuits. *Proceedings of the Sixth Cognitive Science Society Conference*, Boulder, Colorado, pp.337-343.

White, B.Y. & Frederiksen, J.R. (1985). QUEST: qualitative understanding of electrical system troubleshooting. ACM SIGART Newsletter, no. 93, pp.34-37.

Williams,M.D., Hollan, J.D. & Stevens, A.L.(1981). An overview of STEAMER: an advanced computer-assisted instruction system for propulsion engineering. *Behavior Research Methods and Instrumentation*, 13 (2), 85-90.

Why Do Cheetahs Run Fast?
Responsive Questioning in a Case-Based Teaching System

Daniel Choy Edelson
Institute for the Learning Sciences
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, IL 60201

Abstract: A Case-based Teaching (CBT) system consists of two components, a challenging, naturally motivating *task environment*, and a *storyteller* which monitors the student's actions within the task environment looking for opportunities to present instructive cases. CreANIMate is a CBT system that uses video to teach children about animals in the wild. The CreANIMate system explores the roles of stories and of question-asking in teaching.

Introduction

This paper is about using stories and questions to teach. It describes a system called CreANIMate that uses stories as a medium of communication and questions as an organizer and motivator of learning. The CreANIMate program is an example of a class of instructional systems called Case-based Teaching[1] (CBT) systems. Case-based Teaching combines the context provided by an engaging task with the power of stories to convey important lessons. A CBT system consists of two interdependent components, a *task environment* and a *storyteller*. In a Case-based Teaching system, the student engages in a naturally motivating, challenging task. The tools that enable the student to perform that task are provided by the *task environment*. Typically, a task environment consists of either a simulation, a problem-solving environment, or an interactive dialogue. While the student is interacting with the task environment, the *storyteller* monitors the student's actions looking for opportunities to present stories to the student that will help him or her to learn. In the systems currently under development at the Institute for the Learning Sciences, stories consist of brief video clips stored on videodiscs. These stories can take the form of advice from experts, narratives describing personal experience, or footage depicting actual situations from the domain under study (for more detail, see Schank, 1991a).

Case-based teaching is well suited to computer-based educational systems because of the computer's ability to search and retrieve within very large story-bases very quickly (Schank & Edelson, 1990). However, building such systems is both an art and a science. The art of building a case-based teaching system consists of designing a clever environment to engage the student; the science involves indexing a library of cases to enable the system to produce stories at the moment a student will best be able to learn from them. In this paper, I discuss storytelling and question-asking, two key elements of the CBT system CreANIMate.

The CreANIMate System: An Overview

The CreANIMate program is a system currently under development that is designed to teach elementary school children about animals and how they survive in the wild. The program invites a student to create a new animal by taking an existing animal and modifying it in some way. For example, a student might request a gerbil with large claws. The student's task was selected for its inherent appeal and because it encourages creativity. The program initiates a dialogue in which various aspects of the student's proposed animal are considered. The program might conduct a discussion of why it might be helpful for a gerbil to have large claws, or how gerbils use the paws that they currently have. The discussion is accompanied by video of actual animals in the wild that illustrate relevant principles. Thus, the program might show video clips illustrating how different animals use their claws or how certain types of claws are especially well suited for specific purposes. The lesson that underlies the entire interaction is the relationship between the physical features of animals and the ways in which they use those features to help them survive in their environment. The particular examples of these basic principles that any student sees depend entirely on the individual interests expressed by the student.

Thus, the *task environment* in the CreANIMate system consists of a design interaction in which a student proposes a new animal and then explores various aspects of that design. The *storyteller* monitors the student's dialogue looking for opportunities to present video clips showing animals in their natural environments (for more about the interaction between these two modules, see Edelson, 1991).

Aesopic Teaching[2]: Why Teach with Stories?

Schank (1991b) has argued that stories play a fundamental role in organizing human memory. Certainly stories play a central part in communication. If someone were to ask me what my father is like, I would probably respond with a generalization and a story to support it. Why? First, a generalization is not satisfying on its own. A generalization has room for many interpretations, which a story can disambiguate. For example, if I were to tell you that my father is generous, you would not feel like you know very much about him at all. If I told you a story about how he put himself at great financial risk in order to help a close friend, you would feel like you knew quite a bit more about him. Second, stories give people an opportunity to draw their own conclusions. People don't trust the generalizations drawn by others. Finally, a good story has an element of memorability that a generalization doesn't have. It is precisely what differentiates a story from a generalization–details–that enable an individual to recall a story and reuse it in the future.

The central role of stories in communication has a functional justification, also. Humans evolved in a way that enables them to learn from experience. The advent of language meant it was no longer necessary for one person to have experienced a particular situation in order to learn from it. Yet, a mind that evolved to learn from experience would still learn best from communication that comes closest to the actual, firsthand experience of the situation. Since a story bears a closer resemblance to experience than a generalization, people are more receptive to learning from stories than from generalizations. Thus, teaching with stories comes naturally. Parents teach children to behave responsibly by telling them fables, coaches prepare their teams by describing key situations in games from the past, business schools prepare MBA's by working through paradigmatic cases taken from recent corporate history.

In addition, teaching with stories is supported by the theory of *case-based reasoning*. The theory of case-based reasoning (Schank, 1982; Kolodner, 1985) claims that in complex situations, people are unable to take all of the complex features into account in order to reason from first principles. Instead, they use certain key features of a situation in order to recall a previous situation that bears an important similarity to the current situation. They then draw conclusions about the current situation based on its similarities to and differences from the recalled previous *case*. Thus, in a complex decision such as whether or not to invade Kuwait, one might have considered previous cases such as the U.S. involvement in Vietnam and Panama.

In order to perform case-based reasoning, it is important to have a large library of cases that cover an adequate variety of situations. One of the goals of natural human learning is to accumulate such a library. For example, one of the things that people do when they start a new job is to observe their boss as much as possible. They do so in order to build a case library of situations in which their boss has acted, so that they will be able to predict in future situations how the boss will act. However, having a large library with wide coverage, in and of itself, is not sufficient. It is also necessary to have the case library organized, or *indexed*, so that appropriate cases can be retrieved at the moment when they will be useful.

Since stories are a natural form for receiving and conveying information, and they provide important support for case-based reasoning, teaching with stories makes good sense. From the point of view of teaching with stories, there are three important objectives:

- **Coverage.** To learn a subject effectively, the student must be exposed to stories that achieve sufficient coverage of the domain.
- **Organization.** The stories must be presented in a manner that assists the student in organizing his memory of the stories in a way that will enable him to reason effectively about the domain in the future.
- **Memorability.** The stories should be as memorable as possible to maximize their long-term impact on the student.

In constructing a Case-Based Teaching system, coverage and memorability are achieved by careful story collection and selection. Organization is provided by coupling a storyteller with a task environment that presents stories in a context that helps a student organize his learning. In the CreANIMate system, *questions* play a key role in providing organization for the student.

Responsive Questioning: Teaching with Questions

In a system that teaches with stories, the most important issue after "What constitutes a good story?" and "Which stories stories should we use?" is, "How do we establish a context for presenting a story that will maximize the student's learning?" In a Case-based Teaching system the task environment provides the context for stories. A story is presented to the student at the moment when it is directly relevant to the student's current situation in the task environment. Suppose a student using CreANIMate were to add small wings to a cow in an attempt to make it fly. The storyteller might intervene at that moment by presenting a video clip about the physics of flight. This approach is an example of Papert's (1980) *power principle:* that information is learned most effectively when it is presented at the precise moment at which the learner can use the information. According to the theory of failure-driven learning (Schank, 1982), a person is in the best position to learn when some expectation he or she has about the world is violated. A Case-based Teaching system is designed to capitalize on situations in which a student reveals he or she has a misconception or lacks a critical item of knowledge by immediately intervening with an instructive story.

In the CreANIMate system, question-asking is one of the devices that the program uses to establish context for the student. In an educational technique dubbed Responsive Questioning (Schank, 1990), the system uses probing questions to help a student explore the ramifications of his ideas and to introduce instructive stories. Consider the following dialogue from a current version of the CreANIMate system[3]:

Suppose you could create any animal you wanted by taking an existing animal and changing it some way. What would you make?
STUDENT> a tortoise that can run fast
That's an interesting idea. Why might it be useful for your tortoise to run fast?
STUDENT> I dont know
Cheetahs run fast. Do you know why cheetahs run fast?
STUDENT> Show me a video.
VIDEO: cheetah pursuing prey
Now, do you know why cheetahs run fast?
STUDENT> So they can catch their prey.
Yes, that is right. Cheetahs run fast to pursue their prey.
...

In the example above, questions are used in two ways: to organize the dialogue and to introduce stories.

Questions as the Organizers of Dialogues

As soon as the student proposes a new animal, the program poses a question, in this case, "Why might it be useful for your tortoise to run fast?" This question forms the basis for the ensuing dialogue that helps the student to select a reason for his tortoise to run fast. The questions that the system asks are designed to expose basic principles of animal biology and are the types of questions that a student should learn to ask himself when presented with a novel animal. Essentially, these questions represent the curriculum of the system, a set of basic principles that are examined in the context of many different animals. The system teaches by asking these sorts of questions because, "much of what a student needs to hear about an idea that he has had is a question that will make him elaborate upon that idea in some way." (Schank, 1990; p. 15) The strength of the system stems from the fact that the question is asked in direct response to a novel proposal of the student, one to which the student is actively commited. The "responsive question" that the system poses establishes a structure for the dialogue, and hence a context that will help the student integrate new information.

Questions as Motivators and Context Providers

The second use of questions in the example above is to introduce stories. In a Case-based Teaching system, context is established partially by the student's current situation within the task environment. However, in many cases it is also necessary to introduce the story by creating a *bridge* that connects the situation to the story (Bareiss et al, 1991). In

CreANIMate, posing introductory questions serves as a bridge. The story about cheetahs running fast is introduced with the question, "Do you know why cheetahs run fast?" This question serves as a motivator, to draw the student's interest to the story, and as a contextualizer, to help focus the student on the relevant aspects of the story and to provide him with an organizational framework for integrating the story into his memory. The questions that introduce stories motivate a student to attend to the story by evoking curiosity. They help establish a context for learning by providing the student with a basic representational framework for the domain.

The first function of questions that introduce stories in CreANIMate is to motivate the student to learn from the story. A great deal has been written about the value of various types of motivators for learning. (e.g., Lepper & Green, 1979; Malone, 1981) Yet, one motivator, curiosity, never seems to receive the attention it deserves. People are naturally inquisitive, children especially. But what is curiosity? Curiosity is nothing more than an open question. The human mind is set up in such a way that an unanswered question has important status. The power of open questions is demonstrated by the universal experience of letting one's mind wander, and finding it persistently returning to the same unanswered questions. A large quantity of learning is motivated simply by *curiosity*, and the simplest and surest way to evoke curiosity is with a question that is rooted in something that the learner is already interested in. In the CreANIMate system, questions are used to evoke curiosity in the learner by starting with the student's own proposal for an animal, and asking questions that are directly relevant to some aspect of that animal.

The second function of the questions that introduce stories in the CreANIMate system is to establish a context that helps the student to focus on the relevant aspects of a story, and helps him or her to establish an appropriate structure for remembering that story. Schank (1986; see also Ram, 1989) has argued that a person's understanding of any given situation is shaped by the open questions that a person has at the moment they experience that situation. The obvious consequence of that theory is that the same situation experienced with different unanswered questions will be perceived and remembered in a very different way. Psychological studies have shown this effect quite convincingly. In Anderson and Pichert (1978), subjects were presented with a story in which the contents of a house were described. In one group, the subjects were told to imagine themselves hearing the story from the vantage point of a burglar, and in the other group, the subjects were told to put themselves in the position of someone looking to purchase a home. Thus, the two subject groups were provided with quite different sets of open questions. The results, as one would expect, revealed that subjects remembered details that corresponded to the unanswered questions appropriate to their experimental condition. Subjects in the homebuyer condition remembered details better that answered the questions a homebuyer would have when they viewed a house.

The CreANIMate system takes advantage of the role of questions in understanding to help focus students. Given the theory of case-based reasoning described above, it is important to help the students recognize the appropriate features with which to index the story when integrating it into their memories. The introductory question brings out the features that are significant in the current context and primes the student to focus on those aspects when viewing and remembering the story. The question, "Do you know why cheetahs run fast?" helps focus the student in particular on the cheetah's objective when he or she views a video showing a cheetah pursuing its prey. A video contains a large quantity of complex information, but a student who is watching a video in response to a particular question is primed to attend more carefully to details that will help to answer that question.

In addition, the question helps the student to establish a structure for the information he or she is absorbing. The question tells the student that the relationship between an action, running fast, and the higher level behavior that it enables, pursuing prey, is an important relationship; in general, it tells the student, actions should be considered with respect to the high level behaviors they enable, and in particular, this structure can be used to store information about cheetahs.

Conclusion

The CreANIMate system is designed to demonstrate the power of stories and questions for teaching. Questions motivate the learner and help to establish a framework that the student can use to represent the domain. The stories, in turn, serve as concrete, vivid examples that support and elaborate that framework.

Notes

1. The term *Case-based Teaching* has been used to describe any type of teaching that makes use of cases (Cognition and Technology Group, 1990; Gragg, 1940). In this paper, CBT refers to a specific class of teaching systems described by Schank (1991a).
2. Bill Ferguson, also of the Institute for the Learning Sciences is the originator of the expression *Aesopic teaching.*
3. This transcript has been edited for length and readability. The student enters a response by selecting from multiple choices and by filling in blanks in partial sentences, e.g., In response to, "Do you know why cheetahs run fast?" the student can select, "So it can...", "Show me a video." or "Skip this."

Acknowledgements: I wish to thank Bill Ferguson for helpful comments on an earlier draft of the paper. The CreANIMate team includes, in addition to the author, William Fitzgerald, Riad Mohammed, David Newton, Diane Schwartz, and Ian Underwood .

This work is supported in part by a grant from the IBM Program for Improving K-12 Education Through Innovative Uses of Information Technology, and by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of the Arthur Andersen Worldwide Organization.

References

Anderson, R. C. and Pichert, J. W. Recall of previously unrecallable information following a Shift in Perspective. *Journal of Verbal Learning and Verbal Behavior* 17, 1-12.

Bareiss, R., Ferguson, W. and Fano, A. (1991) The story archive: A memory for case-based tutoring. *Proceedings of the DARPA Case-Based Reasoning Workshop,* Morgan Kaufmann Publishers: San Mateo, CA, (in press).

Cognition and Technology Group at Vanderbilt, The. (1990). Anchored instruction and its relationship to situated cognition. *Educational Researcher,* 19 (6), 2-10.

Edelson, D.C. (1991). Oh, the stories I could tell: Managing an aesopic teaching dialogue. *Proceedings of the DARPA Case-Based Reasoning Workshop,* Morgan Kaufmann Publishers: San Mateo, CA , (in press).

Gragg, C. I. (1940). Because wisdom can't be told. *Harvard Alumni Bulletin*, 78-84.

Kolodner, J. L., Simpson, R. L. and Sycara-Cyranski, K. (1985). A process model of case-based reasoning in problem-solving. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence.*

Lepper, M.R., & Greene, D. (1979). *The hidden costs of reward*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Malone, T. (1981). Toward a theory of intrinsically motivating instruction. *Cognitive Science*, 4, 333-369.

Papert, S. (1980). *Mindstorms: children, computers, and powerful ideas*. New York: Basic Books, Inc.

Ram, A. (1989) Question-driven understanding: An integrated theory of story understanding, memory, and learning. PhD Thesis, Yale University, New Haven, CT.

Schank, R.C. (1982). *Dynamic Memory*. Cambridge: Cambridge University Press.

Schank, R.C. (1986). *Explanation patterns: Understanding mechanically and creatively.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C. (1990) Teaching architectures. Institute for the Learning Sciences Technical Report #3, Northwestern University.

Schank, R. C. (1991a). Case-Based Teaching: Four experiences in educational software design. Institute for the Learning Sciences Technical Report #7, Northwestern University.

Schank, R.C. (1991b). *Tell me a story : A new look at real and artificial intelligence.* New York: Simon and Schuster.

Schank, R. C. and Edelson, D. J. (1990). A role for AI in education: Using technology to reshape education. *Journal of Artificial Intelligence in Education* 1(2): 3-20.

Continuous Analogical Learning

Thomas C. Eskridge
Department of Computer Science & Computing Research Laboratory
New Mexico State University, Las Cruces, NM USA 88003-0001

Abstract: We have developed a theory of human analogical reasoning and learning called Continuous Analogical Reasoning (CAR) which provides a basis for developing computational models of human reasoning and learning by analogy. CAR places emphasis on the interactions among the three stages of analogical reasoning. We have developed an implementation of CAR called ASTRA. ASTRA uses hybrid connectionist-symbolic knowledge representation and processing mechanisms to model the CAR theory. We are currently evaluating CAR and ASTRA as models of human analogical reasoning and learning.

Introduction

Analogy is ubiquitous in learning. It is traditional pedagogical methodology to explain a new topic in terms of an example from a different, but more well understood, domain. For example, the famous Rutherford atom analogy, "The atom is like our solar system," explains the unfamiliar atom in the more familiar terms of the solar system. Thagard *et al.* (1989), discussed a number of cases of learning by analogy in the domain of chemistry. An example from their study is "The flow of energy through molecular motion in a gas is like the flow of wealth in a population." In these instructional statements, the purpose of the analogy is to use existing similarities between the two domains to entice further similarities that will add structure to the unfamiliar target domain.

It is often the case that students are confronted with complex devices or environments in which they are asked to "figure it out." This occurs commonly in scientific reasoning where the system of interest may have a large number of hidden aspects. This approach is often taken in the "microworlds" approach to CAI, where students are given a computer environment to experiment with and "discover" new information (Papert, 1987). In such cases of "instructionless" learning or learning by discovery, analogy plays a role by allowing the student to hypothesize information to be true in the target domain based on analogical mappings to a source domain the student is familiar with. Analogy also provides a means for learning by reinterpreting the target domain in terms of the source. In this case, the entire conceptualization of the target domain may be radically altered in order to more closely resemble the source domain. Once this restructuring is done, traditional learning methods or analogies can be used with the modified target.

Given these modes of learning using analogy, it is important to understand the analogical reasoning and learning process in and of itself. We have developed a theory of human analogical reasoning called Continuous Analogical Reasoning (CAR) which provides a basis for developing computational models of analogical reasoning (Eskridge, 1989). CAR is unique in that is proposes that the interactions among the three stages of analogical reasoning play an important role in the analogical reasoning process. Holyoak and Thagard (1989) and Thagard *et al.* (1990) identify three types of constraints acting within the stages of analogical mapping and retrieval: syntactic, semantic, and pragmatic. Our work extends theirs by identifying interactions *between* stages that can also be classified in terms of syntactic, semantic, and pragmatic constraints. This extension also makes constraints directional, having an initiating stage and a recipient stage, and allows the products of each stage themselves to be thought of as constraints (see Fig. 1).

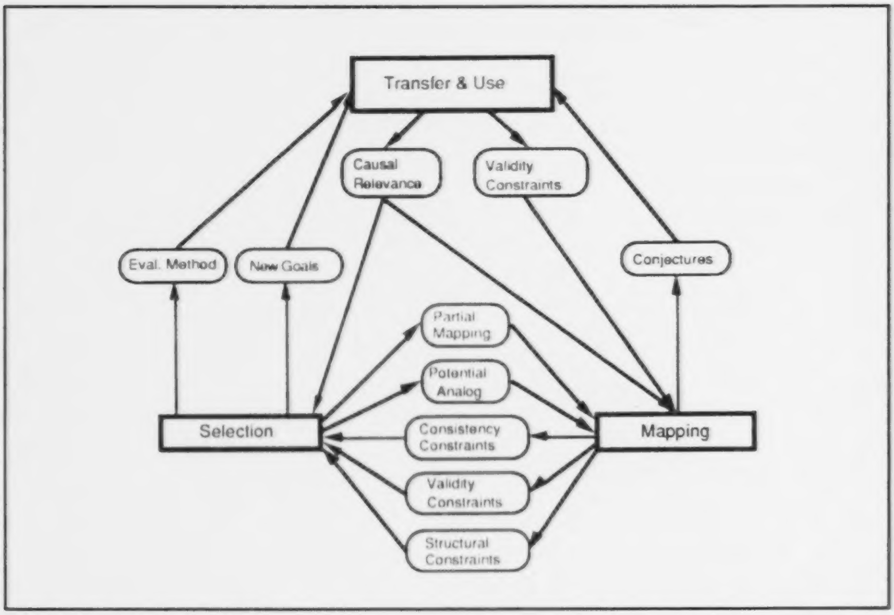

Transfer & Use
Causal Relevance
Validity Constraints
Eval. Method
New Goals
Conjectures
Partial Mapping
Potential Analog
Selection
Consistency Constraints
Mapping
Validity Constraints
Structural Constraints

Figure 1. Interactions between stages of analogical reasoning showing directional constraints acting between stages.

ASTRA: A Computational Model of Continuous Analogical Reasoning

The ASTRA system is an initial attempt to model the CAR theory. In the design of this program, we were influenced by the task decomposition and processing requirements identified in our study of the human analogical reasoning literature. However, we are also interested in ASTRA's performance as a case-based reasoning system, and tried to make ASTRA as generally applicable as possible.

ASTRA uses hybrid connectionist - symbolic knowledge representations and processing mechanisms to implement the CAR theory. The knowledge representation is very similar to a structured localist connectionist network. Nodes represent concepts or microfeatures that are shared between experiences and related to other concepts by weighted links. The representation differs from this in that the links are labeled and inspected during processing. Also, ASTRA requires dynamic link creation during processing.

The knowledge used by ASTRA can be divided into four parts: The hierarchy of concepts, the story representation, rules, and the process-created data structures. The hierarchy defines the concepts that are known by the system and that are used to describe the experiences of the system. The story representation ties together concepts found in the hierarchy with relational information, forming a conceptual representation of the actors and actions of a story. Rules are represented as links in the hierarchy of concepts. The process-created data structures are structures created in response to the actions taken by the selection, mapping, and transfer & use processes. There are two such data structures: markers and bridges. A marker is simply a pointer to the node originating the mark. A bridge is a special type of link that has pointers to other bridges that it uses or that use it.

There are four basic processing mechanisms used in ASTRA to implement the CAR theory: spreading of activation, marker passing, directed retrieval, and traditional symbolic procedures. Automatic spreading of activation passes activation from a node to all neighboring nodes. A user specified decay rate attenuates the activation as it is spread in the network. The marker passing mechanism follows the activation spreading process and puts markers containing information about the origin of the activation on each node receiving activation. When a marker is placed on a node from a different story representation, a bridge link can be created between it and the target node. The directed retrieval process complements the above two processes in that it allows activation and markers to be spread between nodes only over specified links. The symbolic procedures used include procedures for creating bridge links and for determining untouched portions of story representations. More detail on the processing mechanisms can be found in (Eskridge, in prep).

Modelling Human Analogical Reasoning

There are a number of questions that a model of human analogical reasoning should be able to answer: Why is one source favored over another? How are correspondences created between the objects and relations in the target domain and those in the source domain? Of all the possible conjectures sanctioned by a particular mapping, which ones should be transferred to the target domain? One of the methods used to evaluate models of human reasoning and learning is the determination of the degree to which the answers to these questions, and other questions like these, conform to the current results in the psychological literature. We now look at the answers proposed by CAR for the above three questions, and discuss how ASTRA implements those answers.

Why is one source favored over another? The CAR theory makes a number of predictions about retrieval that are consistent with much of the psychological research on analog retrieval. Briefly, CAR predicts that in problem solving contexts, people will tend to retrieve analogs that have a high degree of semantic similarity, that are sensitive to the structure of the target, and that will be relevant to the current goals of the system.

Semantic similarity between the target representation and the representations of the source analogs in memory play a major role in analog retrieval (Gentner and Landers, 1985; Gick and Holyoak, 1980, 1983; Holyoak and Koh, 1987). In ASTRA, the selection process spreads activation and passes markers through the micro-features and localist concept nodes of memory. Concepts in ASTRA are semantically similar if they are identical, if they have a common ancestor in the hierarchy, or if they are in close organizational proximity. Concepts receiving activation from the selection process are considered similar.

Syntactic constraints in the form of structural similarity have also been found to play a role in analog retrieval (Gentner and Landers, 1985; Holyoak and Koh, 1987). In ASTRA, structural similarity is measured by summing the activation traces on the special "bridge" links created during the selection process, connecting a target node and a node in a different analog representation. When bridge links are created, the activation passing over the new bridge is added to the activation of any older bridge that lies in the path taken by the activation spread between the two ends of the new bridge.

Subjects in experiments have shown to be sensitive to the goals of their given tasks in analog retrieval (Faries and Reiser, 1988; Seifert *et al.* 1986). ASTRA uses the goals of the system to enhance the activation strength of concepts related to those goals. A typical case is determining the effects of a particular action. Any concept related to a concept in a potential source analog by a "cause" relation will have the activation passing over that relation's link augmented. Because of the augmentation, activation can now travel further than unaugmented activation. This means that the similarity between the target and potential source will be modified based on pragmatic constraints.

How are correspondences created? The CAR theory proposes that much of the work in setting up correspondences between the objects and relations in the target and those in the source analog is accomplished during the retrieval stage. As described above, much of the work done to select a source analog involves comparison of the analog representations. The CAR theory postulates that the comparisons done in the retrieval stage create a "seed" set of object and relation correspondences which must be extended by the mapping stage to encompass as many elements of the target as possible. CAR predicts that the extended mapping will be strongly influenced by structural similarity, that semantically similar entities will be more likely to correspond than semantically unrelated entities, and that entities that are functionally relevant (Keane, 1989) to the current goals are likely to correspond.

Studies have shown that structural similarity is directly related to ease of mapping (Gick and Holyoak, 1980; Holyoak and Koh, 1987; Ratterman and Gentner, 1987). The mapping stage is affected in the same way that the retrieval stage in ASTRA is influenced by structural constraints. To extend the "seed" mapping, the mapping process initiates directed retrieval of objects in the target domain using relations between unmapped and mapped source nodes. If the relation is present in the target, a bridge is created between the unmapped source and the target node. If the relation is not present, it is identified as a possible conjecture.

Studies have shown that the semantic similarity is directly related to ease of mapping (Gentner and Toupin, 1986; Holyoak and Koh, 1987; Keane, 1989). In ASTRA, semantically similar entities will be more likely to correspond due to the relatively high activation strength of the bridge connecting them. The high strength of the bridge is due to the short distances activation will have to travel between semantically similar concepts.

Theorists have emphasized the use of pragmatic knowledge in determining correspondences between analogs (Holyoak, 1985; Kokinov, 1990). Some have proposed causal relevance as a key for determining correspondences (Hesse, 1966; Winston, 1980). Others have focused on high-level goals and plans to aid in the determination of correspondences (Carbonell, 1986; Keane, 1989; Seifert, 1988). In ASTRA, goal-related information influences the mapping stage by preferring that entities are placed into correspondence when they are related by causal relations in both the target and the source.

Which conjectures are selected for transfer to the target? The CAR theory proposes that out of all possible conjectures sanctioned by a mapping, the ones that will be transferred depend on the task context of the reasoner. In learning domains, it may be appropriate to transfer all possible conjectures, since the task is to acquire as much information about the target domain as possible. However, in problem solving contexts, only as many conjectures as necessary to solve the current goals of the system need be transferred.

Approaches to determining what to transfer generally fall into either the pragmatic or syntactic camps. The pragmatic approach represents the idea that the context of the problem and the goals of the reasoner are the main constraints on what is transfered (Carbonell, 1986; Holyoak and Thagard, 1989; Kokinov, 1990). This approach will transfer only the amount of information necessary to enable the reasoner to carry on with problem solving or solve the goal outright. The syntactic approach is best exemplified by the Structure Mapping Theory (SMT) (Gentner, 1983). The structure mapping theory suggests that transferred information conforms to the *systematicity* principle: relational properties that are part of a system of relations are more likely to be transferred than are isolated relations.

The determination of what to transfer in ASTRA is based on the goal-relatedness and the activation strength of the conjectures proposed. In problem solving contexts a conjecture is selected for transfer if it is directly goal-related, e.g., it involves a "cause" relation. A conjecture may also be transferred if its activation is above a user-specified threshold. In learning contexts, the transfer threshold is set to zero, enabling all conjectures to be transferred to the target.

Conclusions

In this paper we have presented the Continuous Analogical Reasoning theory as a basis for developing computational models of human analogical reasoning. We presented ASTRA, an initial attempt at developing a computational model of analogical reasoning and learning based on the CAR theory. To demonstrate the efficacy of CAR and ASTRA as cognitive and computational models of analogical reasoning, we proposed solutions to three key questions that must be addressed by all theories and models of analogy. We have found that the performance predictions made by our theory has considerable psychological support, and that our computational model adequately models these results.

References

Carbonell, J. (1986). Derivational Analogy: A theory of reconstructive problem solving and experience acquisition. In R. Michalski, J. Carbonell, and T. Mitchell (Eds.) *Machine Learning, Vol II*, Morgan Kaufmann.

Eskridge, T.C. (in prep) A Hybrid Model of Analogical Reasoning, manuscript in preparation.

Eskridge, T.C., (1989). Principles of Continuous Analogical Reasoning. *Journal of Theoretical and Experimental Artificial Intelligence*, 1(3): 179-194.

Faries, J. M. and Reiser, B. R. (1988). Access and use of previous solutions in a problem solving situation. In *Proceedings of the Tenth Annual Meeting of the Cognitive Science Society*, Cognitive Science Society, pp. 433-439.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science7*, (2): 155-170.

Gentner, D. and Landers, R. (1985). Analogical Reminding: A good match is hard to find. In *Proceedings of the International Conference on Systems, Man and Cybernetics*, IEEE, pp. 607-613.

Gentner, D. and Toupin, C. (1986). Systematicity and surface similarity in the development of analogy. *Cognitive Science 10*: 227-300.

Gentner, D. (1988). Analogical Inference and Analogical Access, in A. Preidetis (Ed.) *Analogica*, Pitman Publishing.

Gick, M.L. and Holyoak, K.J. (1980). Analogical problem solving. *Cognitive Psychology 12:* 306-355.

Gick, M.L., and Holyoak, K.J. (1983). Schema Induction and Analogical Transfer. *Cognitive Psychology 15*(1): 1-38.

Hesse, M.B. (1966). *Models and analogies in science.* Univ. of Notre Dame Press.

Holyoak, K.J., (1985). The pragmatics of analogical transfer, in G.H. Bower (Ed.), *The Psychology of Learning and Motivation 19.*

Holyoak, K.J. and Koh, K. (1987). Surface and structural similarity in analogical transfer. *Memory & Cognition 15*: 332-440.

Holyoak, K.J. and Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science 13:* 295-355.

Keane, M.T. (1989) *Analogical Problem Solving*, Ellis Horwood Limited.

Kokinov, B. (1990). Associative Memory-Based Reasoning: Some Experimental Results, in *Proceedings of the 12th Annual Conference of the Cognitive Science Society*, pp. 741-749.

Papert, S. (1987). Microworlds: Transforming Education, in R.W. Lawler and M. Yazdani (Eds.) *Artificial Intelligence and Education, Vol 1.*, pp. 79-94.

Ratterman, M. and Gentner, D. (1987). Analogy and similarity: determinants of accessibility and inferential soundness, in: *Proceedings of the Ninth Annual Meeting of the Cognitive Science Society*, pp. 23-35.

Seifert, C.M. (1988). A Hybrid Model for Controlling Retrieval of Episodes, in *Proceedings of the Tenth Annual Meeting of the Cognitive Science Society*, pp. 731-737.

Seifert, C.M., McKoon, G., Abelson, R., and Ratcliff, R., (1986). Memory connections between thematically similar episodes *Journal of Experimental Psychology: Learning, Memory, and Cognition 12*: 220-231.

Thagard, P., Cohen, D.M., Holyoak, K.J., (1989). Chemical Analogies: Two Kinds of Explanation, in *Proceedings IJCAI-89*, pp. 819-824.

Thagard, P., Holyoak, K.J., Nelson, G., and Gochfeld, D., (1990). Analog Retrieval by Constraint Satisfaction. *Artificial Intelligence 46:* 259-310.

Winston, P.H. (1980). Learning and Reasoning by Analogy. *Communications of the ACM 23*(12): 689-703.

Knowledge-Based Tutoring Systems: Changing the Focus from Learner Modeling to Teaching

Richard G. Feifer and Michal S. Soclof
Institute for the Learning Sciences
Northwestern University
1890 Maple Avenue, Evanston, IL 60201, USA

Abstract: The most common knowledge-based approach to educational software is an intelligent tutoring system (ITS). Unfortunately, ITS research has become so involved with the problem of learner modeling, that the question of how the system would actually teach is usually neglected. We propose a case-based tutoring approach that focuses on teaching rather than learner modeling. We have implemented our approach in VICTOR (Voice, Image and Courtesy tuTOR), a system that teaches effective telephone communication skills.

In a case-based tutoring system, a model of the task allows the system to know when the learner has failed. The system then indexes that failure to one of the cases in its case library. There are three benefits in presenting an appropriate case at the moment the learner has failed: the learner is motivated to listen to the case, the information is relevant, and it is easier for the learner to integrate the case correctly into his own memory.

VICTOR engages the learner in a phone conversation with a simulated caller. The dialogue proceeds depending on how well the student performs. VICTOR diagnoses problems in the dialogue and chooses appropriate stories to tell using only a model of the task; there is no model of the learner.

Knowledge-Based Tutoring

In the typical computer-based tutor, a student learns by practicing on a series of multiple choice tests. The student is given a situation and, based on his current knowledge, must select an answer from a menu of choices. Unfortunately, there are few problems in the real world which are structured like problems on multiple-choice tests. In other words, we rarely encounter situations where we are given a task, several possible ways in which to perform it, and are asked to select the correct solution.

A better computer-based tutor would recognize that there are many actions that any one person can take to complete a task. This tutor would teach by providing an appropriate response to every possible learner action. The student would attempt to accomplish the task, receiving instruction only after he has failed or asked for help. Our goal is to build such a computer-based tutor.

Unfortunately, to program a computer to anticipate every action that any learner could do and provide an appropriate reaction is impossible in most situations. One possible solution is a knowledge-based system (Wenger, 1987). In conventional computer-based instruction, all of the decisions that the system could possibly make are explicitly programmed into the computer. In a knowledge-based system, however, the knowledge needed to make the decisions is represented. The system then uses this knowledge to make decisions. A knowledge-based system is thus able to make inferences beyond what is explicitly programmed. This allows the system to recognize and respond appropriately to actions that the programmer did not anticipate. Therefore, a knowledge-based approach to educational software provides the potential for building computer-based tutors that allow, recognize, and respond appropriately to a wider range of learner actions.

Intelligent Tutoring Systems

The most common knowledge-based approach to educational software is an intelligent tutoring system (ITS). The idea in building an ITS is to represent the three kinds of knowledge a human tutor uses to teach: knowledge of the domain being taught, knowledge of how to teach, and knowledge about the learner. There is no script directing the tutoring. Theoretically, the ITS is able to automatically generate a tutorial interaction using these three knowledge sources.

Sherlock, for example, is an ITS designed to teach people to structure knowledge as they read (Feifer, 1989). The learner's task is to read some text and then create a graphic map representing the ideas in that text. Sherlock uses its representation of the text being read and its representation of a strategy for creating a graphic map to model what the learner was thinking at each step. If the learner's action is judged as wrong, Sherlock's tutoring strategies component generates explanations intended to clarify the learner's misconception which led to the wrong action.

Sherlock was shown to correctly interpret most learners' actions, including those that were unanticipated by the programmer (Feifer, 1990). In addition, Sherlock had some success diagnosing the cause for a wrong action.

Despite these accomplishments, Sherlock was unable to effectively teach. One reason for this shortcoming was that learner modeling, which was central to the project, was so intricate and involved that it became a project in and of itself. Therefore, the design of the project never progressed beyond learner modeling. The issue of how the tutor should teach was never addressed.

Sherlock is not unique in this respect. Most intelligent tutoring systems focus on learner modeling rather than teaching. Some systems (i.e., Corbett and Anderson, 1991) have addressed the issue of feedback, but we have not found a system which has addressed the question, "What would it mean for a computer to teach intelligently?"

Case-Based Approach

Intelligent tutoring systems (i.e., Burton and Brown, 1976; Johnson, 1985) have been successful at modeling the learner. However, a learner model is not a sufficient condition for good teaching. We would also argue that it is not a necessary condition.

We propose a case-based approach to building knowledge-based tutoring systems. As illustrated in Figure 1, the case-based tutor does not have a model of the learner and also does not represent the knowledge needed to accomplish the task being taught. Rather, this model simulates the task. A case-based tutoring system, therefore, avoids the hardest problems inherent in building an ITS. Instead, the student learns by hearing a case (story) that is relevant to a problem he just encountered.

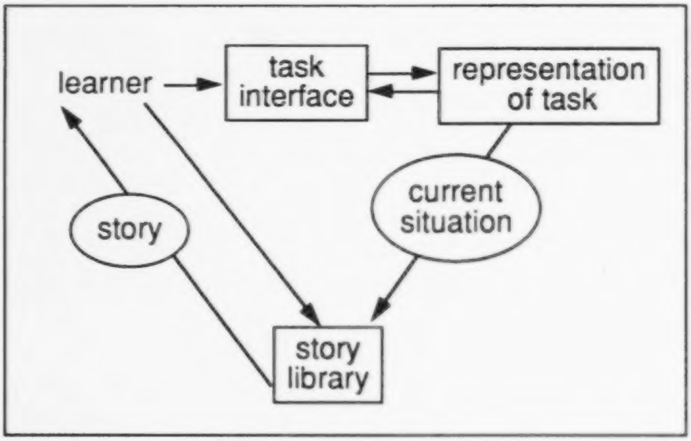


Figure 1. Model of Case-Based Tutor

In a case-based tutoring system, we begin with a simulation of the task we want the learner to accomplish. Simulating the surface features of a task is far easier than representing the knowledge needed to accomplish the task. For example, it is easier to simulate the controls of an airplane than to simulate the mind of the pilot who flies that plane.

The simulation of the task serves two functions: It indicates the results of any learner action, and it produces a high level description of any failure state that can occur. The first function provides the learner with feedback on the appropriateness of any action. The second function lets the system find an appropriate story from the story library to tell when the learner fails.

The story library contain explanations, positive examples and negative examples for the domain being taught. The stories are categorized based on the kind of situation for which they are appropriate. When the learner fails, the system makes no attempt to determine the cause. Rather, the system finds and tells a story whose salient features closely match the situation that occurred in the task.

The right story can help the learner determine what features or actions caused him to fail. It can also help the learner associate more appropriate solutions to the problem that led to the failure. Learning occurs when the learner applies the story to the current situation. He is motivated to try and apply the story because he has just experienced the failure.

One of the hardest tasks in learning is properly indexing new knowledge into memory, so that the new knowledge can be recalled when needed. In a case-based tutoring system, the learner is already thinking about a situation in which the new knowledge is needed before he hears the story. Therefore, in the future, when the student encounters a similar situation he will be able to recall the lessons of the story.

VICTOR

We have implemented the case-based model of tutoring in VICTOR (Voice, Image and Courtesy tuTOR). VICTOR is designed to replace a course on business telephone communication skills offered by Illinois Bell. The goal of the course is to improve the telephone skills of secretaries and operators of Illinois Bell's business customers. Illinois Bell currently provides the course in the form of a four hour class taught by an instructor. Most of the current course is spent listening to the instructor lecture and tell stories. There is little opportunity for practice or feedback.

Using Victor

VICTOR has two main parts. In the first part, the learner can select the role he would like to play. The learner can pick from such roles as console operator, clerk, etc. Based on his selection, a business situation is generated by the computer, and the learner engages in a phone conversation with a computer-simulated caller. The dialogue proceeds depending on how well the student performs. If the learner handles the call poorly, the simulated caller will get angry, frustrated, confused or hang up.

After the dialogue with the computer-simulated caller, the learner proceeds to the second part of the tutor. Here, he listens to his dialogue and evaluates himself. During this evaluation, the learner selects which areas he would like to improve. The tutor then adds to this list any problem areas that it diagnosed. For each of these areas, the student enters a learning module that teaches him and lets him practice relevant skills.

Upon entering a module, the learner watches a short video-taped story introducing the topic. While the video clip is playing, the key points discussed are displayed on a "blackboard." The learner can then request additional information or ask questions by clicking on one of the question buttons (WHY, HOW, HUH, MORE DETAIL). The learner then sees the story that best answers the question.

When ready, the learner can enter the practice part of the module. Here, VICTOR gives the learner tools to let him practice the skill being taught and to evaluate his progress. The learner can practice and receive feedback until he is ready to continue.

Simulating the Task

Since we want to teach the learner how to handle a variety of business calls, we begin by having him handle a simulated caller. Specialized hardware and software were used to create a realistic dialogue. For example, the learner hears the phone ring and the caller's voice using digitized sound files. He then responds by speaking into a microphone. The caller's answers vary in both tone and content, depending on the learner's response. There are separate sound files for the caller's reactions in varying degrees of anger, frustration or satisfaction. VICTOR uses a simple rule system to determine which file should be played next.

The learner must respond to the caller as he would in an actual phone call, by immediately verbalizing his response. There is no time to plan a good answer, since the system starts recording as soon as the simulated caller stops talking. The tutor, however, does not understand natural language. Therefore, in order for the computer to understand what the learner said, he is then asked to reconstruct his answer by choosing from a predetermined list of sub-phrases.

Feedback

VICTOR helps the learner improve his ability to handle phone calls in two ways: providing the learner with feedback, and providing the learner with tools for self-evaluation.

The dialogues provide immediate feedback to the learner since he hears what the results of his action would be in the real world. This is the only form of feedback the learner would normally receive on the job. The difference in VICTOR, however, is that the feedback is non-threatening. The learner is free to experiment with different approaches to handling a call without the risk of losing customers or his job.

In addition to the immediate feedback, VICTOR remembers any problems that occur during a dialogue and adds them to the list of skills the learner needs to study. For example, if the learner does not identify the name of the store, in addition to directing the caller to get aggravated and to ask the name of the store, VICTOR would also place the topic "Greeting the Caller" on the list of skills the student has to learn. The flow of the conversation, however, will not be interrupted at this point. The tutor will mention the problem explicitly only after the dialogue and after the learner has had an opportunity to notice the problem himself.

VICTOR also provides the learner with tools that increase his ability for accurate self-evaluation. These tools give the learner opportunities to look at his performance in ways that would normally not be possible.

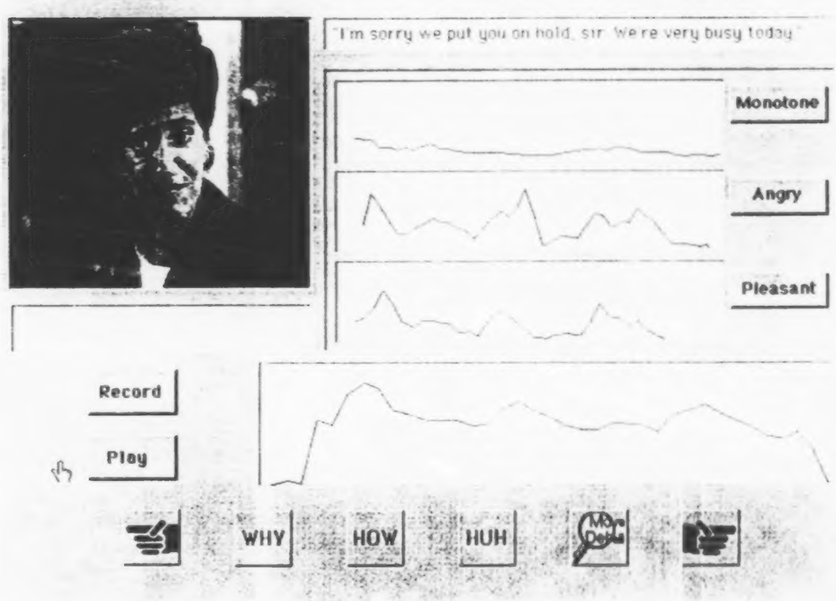
A simple example is the capability that lets the learner play back both sides of a conversation. After a dialogue, VICTOR interleaves the learner's responses with the caller's responses to allow the learner to hear what the entire conversation might sound like from the caller's point of view. In addition, the learner is given a check list of possible problems to consider while listening to the dialogue. Any problems that are chosen are added to the list of skills the learner needs to study.

A practice module with self-evaluation tools is available for each of the ten skill areas that have been identified for this domain. For example, in the practice module for inflection, the learner can listen to and see pitch curves for voice samples that are monotone, angry, or pleasant. He can then record his own voice sample and once the system has analyzed it, listen to and see his pitch curve (Figure 2). The learner can then compare the model voice samples with his own.

Stories

The heart of VICTOR is the story library. To ensure that the stories contain the necessary information and are presented appropriately, the teachers who currently teach the course were recorded on video tape. The teachers were chosen because they are charismatic, know the material, and relate well to the types of students who will be using the tutor. The video tapes of the teachers were then edited into short clips.

For each problem encountered during a dialogue, VICTOR finds and shows a video clip whose salient features best match the salient characteristics of that problem. Once a story has been seen, the learner is able to ask questions that lead to related videos. To provide this functionality, the videos were placed in a network. A link between two videos represents a question the learner might ask of the first video that is answered by the second video. The goal of constructing the network is to allow the learner to see a maximum number of stories, while minimizing irrelevant material. Thus, each video is linked to as many other appropriate videos as possible.



"I'm sorry we put you on hold, sir. We're very busy today."

Figure 2. Practice Module for Inflection

Conclusion

VICTOR has been able to overcome some of the primary obstacles that have prevented intelligent tutoring systems from making an impact on teaching and training.

One obstacle facing intelligent tutoring systems is that they are difficult and time-consuming to build. By eliminating learner modeling, VICTOR overcomes this problem to a large extent. VICTOR, therefore, was a relatively short-term project. It took only seven months to develop a prototype that can be used by students to improve their phone skills.

Another obstacle faced by a conventional ITS is its complex structure. In order for an ITS to teach, it must have a learner model, a representation of the knowledge needed to complete a task and a generator for creating instruction from these two parts. The building of these three components is a difficult AI problem that has not yet been solved. Therefore, intelligent tutoring systems have had difficulty teaching.

VICTOR's teaching is contained in video clips of teachers telling good stories. The system need only find the right story to tell at the right time. This is based on the premise that the best learning takes place after a learner has tried and failed to accomplish a task. If, at that point, the learner is told a story, he will be motivated to listen to the story and be aware of the context in which the story is relevant.

At this point, we cannot make concrete claims regarding VICTOR's teaching effectiveness. An empirical study has just been initiated to evaluate how well VICTOR teaches. It is significant, however, that the system is complete enough to measure how well it teaches.

Acknowledgements: This research was supported by Ameritech, an Institute Partner. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from IBM, the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058 and the Office of Naval Research under contract N00014-90-J-4117, by the Office of Naval Research under contract N00014-89-J-1987, and by the Air Force Office of Scientific Research under contract AFOSR-89-0493.

In addition to the authors, the following people are involved in the development of VICTOR: Greg Sandell, John White, Yat Keung, Leena Nanda, Jim Christoff, Adele Meadows, Ann Jacobs, Diane Schwartz, Jarrett Knyal, Michael Korkuska, and Michael Engber. Special thanks to Leena Nanda for her help with editing the manuscript. We would also like to thank the Illinois Bell instructors, Karen Phillips and Lodita Hicks.

References

Burton, R. and Brown, J.S. (1976). A tutoring and student modeling paradigm for gaming environments. In Coleman, R. and Lorton, P. Jr. (eds.). *Computer Science and Education.* ACM SIGCSE Bulletin, 8, 236-246.

Corbett, A.T. and Anderson, J.R. (1991). Feedback control and learning to program with the CMU Lisp Tutor. Paper presented to the American Educational Research Association, 1991 Annual Meeting, Chicago, IL.

Feifer, R.G. (1989). An intelligent tutoring system approach to teaching people how to learn. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society,* Ann Arbor, Michigan.

Feifer, R.G., Dyer, M.G., Baker, E.L., Flowers, M., and Read, W. (1985). *Teaching inferencing strategies* (Technical Report UCLA-AI-85-5). University of California, Los Angeles, Computer Science Department, Artificial Intelligence Laboratory.

Johnson, W.L. (1985). *Intention-based diagnosis of errors in novice programs.* New Haven: Yale. U/CSD/Technical Report, #395.

Wenger, E. (1987). *Artificial intelligence and tutoring systems.* Los Altos, CA.: Morgan Kaufmann Publishers.

ASK Systems: An Approach to Story-Based Teaching

William Ferguson, Ray Bareiss, Richard Osgood, and Lawrence Birnbaum
The Institute for the Learning Sciences
Northwestern University, Evanston, Illinois 60201

Abstract: Telling stories is an effective way to teach aspects of nearly every task or domain. Ideally, students would learn from stories told in the natural course of conversations with experts in the domain of interest. Unfortunately, experts are scarce, so such direct access is limited. Our solution to this problem is to broaden access to expert stories through the development of hypermedia systems designed to provide an interaction that emulates, as much as possible, the cognitively relevant aspects of a conversation with an expert. We have constructed a number of such story-based teachers, called ASK systems, in domains as diverse as trust bank consulting, Presidential decision-making, and the historical causes of the success of a nation's industries in the global market. In this paper we identify and discuss three primary factors influencing the design of the indexing networks which lie at the heart of these systems: the task being taught, the learning goals of the student, and the source and nature of the stories.

Introduction

Telling stories is an effective way to teach aspects of nearly every task or domain. An appropriate story told in an appropriate setting conveys important information, and, equally importantly, provides contextual cues that facilitate recall of that information in situations where it is likely to be applicable, as indicated by work in both case-based reasoning (Riesbeck and Schank, 1989) and situated cognition (Brown et al. 1989). This observation forms the basis or our work in case-based teaching (Schank, 1991).

Ideally, students would learn from stories told in the natural course of conversations with experts in the domain of interest. Unfortunately, experts are scarce, so such direct access is limited. Our solution to this problem is to broaden access to expert stories through the development of hypermedia systems (see, e.g., Conklin, 1987) designed to provide an interaction that emulates, as much as possible, the cognitively relevant aspects of a conversation with an expert. We have constructed a number of such story-based teachers, called ASK systems (Schank et al. 1991), in domains as diverse as trust bank consulting, Presidential decision-making, and the historical causes of the success of a nation's industries in the global market. A student interacts with an ASK system by navigating through an indexing network to find relevant stories, often presented in the form of short video clips, in a way that is intended to simulate a conversation in which the student asks questions and an expert answers by telling stories.

It is important to make clear that an ASK system contains only a trivial algorithm: The key to its functionality lies in the *content* of the stories contained in the system, and of the indexing network that enables access to those stories. The theoretical basis underlying our approach is provided by *case-based reasoning* (CBR), a paradigm for understanding and problem-solving which takes memory for specific experiences to be the foundation of both human and artificial cognition. Two major teaching principles follow from such an account of human cognition: First, if expertise is best represented in terms of specific experiences, then a natural way to facilitate the assimilation of new information by humans is to present that information in the form of cases or *stories*. Second, because the usefulness of information depends crucially on the ability to retrieve it when appropriate, such cases are best presented in a context that enables the listener to determine where they should be indexed in his or her memory, and how they should be connected to other relevant cases.

This paper discusses the design issues that arise in the construction of ASK systems, based on a comparative analysis of three implemented prototypes. From a pedagogical perspective, the most crucial issues concern what sorts of questions students are likely to ask initially and what sorts of follow-up questions they are likely to ask in response to the stories that are presented. This translates into the problem of designing an indexing network which will enable students to ask, and find answers to, their questions while navigating through a large-scale story database. The analysis reveals three factors which substantially determine how our systems address these issues: the task which the student is learning, the nature of the case base of stories, and the goals of the student.

Interacting with an ASK System

Before elaborating the factors that influence the design of an ASK system, it is useful to consider an example of a student interaction. Suppose that a new consultant is about to begin his first trust banking job. He wants to know what kinds of questions he should be asking and how he should conduct himself to make a good impression on the client. As an alternative to a one-on-one tutorial with an expert, the novice consultant might take his questions to ASK TOM, a system designed to teach trust bank consulting.

When the interaction begins, ASK TOM displays a screen depicting the agents and relationships which characterize the domain of trust bank consulting. The user selects the element from this diagram which is most closely connected with his question. In this example, the user might choose the relationship between an individual consultant and a client trust bank, labelled "Your job on a trust consult," as being most relevant to his query. This results in the display of a screen displaying a number of relevant stories, categorized by general points, or *themes*. Under the theme "Getting answers to your questions" is a list of stories, including one entitled "First things to find out: Customers and assets." If he views this story, he will be told about questions that he can ask that will elicit potentially sensitive information from the client without putting the client on the defensive. Our hypothetical consultant has now finished the process we call *zooming*–navigating from the initial screen of the system to a particular story which answers a question. The screen now displays a number of stories, linked to the current one, which address logical follow-up questions. The links between stories (i.e., the categories of follow-up questions) are of eight possible types in ASK Tom. For instance, after watching the first story about how to find out about customer and asset information, our novice consultant may wonder why this information is useful to him. He can find the answer to this question by looking at the story linked to the current one as providing its **context**. Determining an appropriate set of links to facilitate browsing in this way is a key issue in the design of an ASK system.

Factors in the Design of ASK Systems

To facilitate the interaction between the student and an ASK system, the indexing network used by the system and the goals of the student must remain "in sync", i.e., the options presented by the system should encompass the student's immediate interests and they must be presented in a way that will allow the student to translate his or her questions readily into the system's terms by choosing appropriately. To achieve such a smooth meshing, three factors must be accommodated by the indexing scheme used in an ASK system:

Task. The indexing network must reflect the structure of the cognitive processes–e.g., planning, explanation, diagnosis, etc.–which are involved in the particular tasks confronting the student. To be more specific, cases must be interconnected in ways that facilitate the performance of these cognitive processes, and the information from each story relevant to these processes and tasks must be identified. For example, if the task involves problem solving, then stories that describe problem situations and possible solutions must be connected to other stories about positive and negative consequences of the alternative solutions.

Story base. The nature and sources of the stories to be presented has, as might be expected, a large impact on the indexing network. Ask systems constructed at ILS have used two sources of stories. The first is videotaped interviews which we have conducted with experts. In this case, both the expert and the interviewer understood the pedagogical purpose of the video and biased their interaction to suit the task being taught. The second source of stories has been archives of existing material. In some cases this video (or text) was created for other purposes, not associated with any pedagogical agenda.

Student goals. The anticipated learning goal or motivation of the student also influences the design of the indexing network. If the user of the system is expected to have the goal of performing the task which the system is designed to teach, then the goals of the student will be taken into account along with the task itself. If, on the other hand, the goals of the student are more general than the teaching goals of the system, or if the student's goals cannot be predicted, then the system design must be adapted to entice the student to delve into the material as quickly as possible.

Three ASK Systems and Three Approaches to Indexing

In this section we will discuss how the factors described above have influenced the design of the indexing networks used in three prototype ASK systems which have been constructed at Northwestern University's Institute for the Learning Sciences (ILS).

ASK Tom

The ASK Tom system (Schank et al. 1991) was the first ASK system built at ILS. As described above, its was designed to teach trust-bank consulting to junior consultants. The story data base was drawn from videotapes of three days of interviewing a trust bank consulting expert who knew that he was producing stories for use in a teaching system. The emphasis during the interview was on collecting the "wisdom" of the expert in the form of "war stories" about problems on real jobs, along with brief observations about trust bank consulting as the industry exists now and as it might exist in the near future.

In designing ASK Tom, we could count on the student to have the goal of learning to perform the task of problem solving in trust bank consulting. In addition, ASK Tom's story base was directly oriented towards this task. This is the ideal situation for the construction and use of ASK systems: The task the system is designed to teach, the task the student is trying to learn, and the task the that story base is biased to transmit are all the same. The system needs no implicit agenda to present any particular material since, if the indexing network merely enables the student to gain answers to his or her questions, then the needed information will be presented.

The zooming index for the ASK Tom system consists of two or three layers of nodes arranged into a discrimination net represented to the user in the form of a two- or three-layered menu, as described in the example above. The options presented are either diagrams illustrating some subset of the trust bank consulting world—e.g., the steps in a consulting engagement, or the agents and relationships involved in government regulation—or else sets of lesson points (themes) which pertain to some aspect of trust bank consulting.

The link types used in the browsing index for ASK Tom encompass the types of information useful in problem solving. They are **Warnings, Opportunities, Results, Background, Context, Examples, Indicators, and Alternatives.** These links allow a straightforward categorization of problem solving follow-up questions while at the same time allowing the system to present follow-up information that the student might not have thought to ask about. These link types are based on a theory of plausible topic transition in conversation (Schank, 1977), as well as work on abstract problem-solving knowledge (e.g., Sussman, 1975; Schank, 1982).

ASK the President

The ASK The President system is being constructed to teach contemporary American history to high school students. By interacting with Ask The President, a student will learn how Presidential decisions are driven by events and political considerations and, in turn, how those decisions shape future events. Thus, the task being taught to the student is historical *explanation*.

The case base for ASK The President consists of news video of major events of the last 15 years, accompanied by public statements from the three Presidents and other major political figures of those times. Unlike ASK Tom, the video was acquired from archives rather than being produced expressly to satisfy the pedagogical goals of the system. Fortunately, the material is naturally cohesive: Sequences of events naturally fall under a small number of general topics (such as the Energy Crisis or U.S.-Soviet Relations). In general, there is much higher connectivity within topics than between them, although there are often natural relationships between contemporaneous events (such as the Energy Crisis and the Recession of the 1970s) that must be captured to provide a complete historical picture.

In general, high school students can't be expected to have the goal to perform historical explanation or even to posses an *a priori* motivation to use the system.[1] To meet the needs of these students, the zooming interface of Ask The President is structured to allow the student to select interesting events directly rather than descending through a hierarchical, graphical representation of the domain to find the answers specific questions. Such an approach was chosen because we felt the need to capture the interest of potentially unmotivated students outweighed the need to provide a reference resource of historical information.

When an event is selected, the student enters its chronology at a critical point rather than at the beginning (e.g., at the taking of American hostages rather than the beginning of the Iranian revolution). The browsing interface embodies a general explanatory framework which allows the student to move through time backward (to causes), forward (to results), and laterally (to related events) at will. As his or her interest is captured by particular events, the student is able to delve beyond the news stories to hear the opinions of the President and other decision makers, to hear details of government policies, and to acquire background information necessary for the explanation process. In summary, the browsing links of ASK The President are **causes, results, related events, opinions, policies,** and **background facts.**

ASK Michael

ASK Michael was the first text-based ASK System produced at ILS. It is built around *The Competitive Advantage of Nations* (Porter, 1990). The purpose of the book is to describe those factors that account for the international success of nations in specific industries in terms of a new theory of competitive advantage. The system's purpose is to provide ways of reading the book, that go beyond what is possible in print. There is no particular task orientation. Students browse through the book guided primarily by interests generated by what they have already encountered. We have attempted to maximize the likelihood that questions which arise in the course reading can be answered to the extent that answers are available in the story base.

The story base in our initial implementation consists of several sections from the book (about one-third), which have been segmented into short pieces each with a single main point. As such these segments retain the essence of the pedagogical objectives which the author had in writing the book in the first place. Namely, the task of explanation in the realm of the success and failure of particular industries in particular sets of circumstances. The stories together embody the support for a particular set of factors which can be used to evaluate and make such determinations.

The choices open to users of ASK Michael are developed by a family of zooming interfaces and a single browsing interface. There is no specific theory or model level in ASK Michael to help organize access to the book. This is because the book itself presents the factors and forces that produce competitive advantage. ASK Michael is an example of a system whose story-base itself contains the model being taught. To accomplish the zooming function, ASK Michael contains four distinct views: by story point, by general topic, by topic relationships and by interesting themes. This last method most closely supports the purpose of ASK Michael as an interest-based browsing system. Using a limited taxonomy of intrinsically interesting themes, e.g., "Odd Practices," a student can get started at a point where interest is high.

The browsing interface then takes over. This interface was constructed by analyzing each story's content both for the questions it raises and the questions it answers. Because there is no specific task focus for ASK Michael, these analyses are done from the perspective of questions that the general reader is likely to ask. By matching like questions (raised and answered), a richly interconnected web of stories is produced that reflects a new level of content indexing beyond that found typically in the back of a book. We believe our content analyses of the stories has produced a good covering set of questions for users to employ in keeping their interests alive as the follow a thread through the book.

Conclusion

The basic interface principle underlying the construction of ASK systems is that even a rudimentary AI theory of a task or a domain provides constraints that can be used to structure human-computer interactions in a way that is meaningful to the human user. To the extent that an interaction so constrained corresponds to the user's understanding of the task and the domain, the context provided by the system at any given point will agree with the context that the user naturally expects. This shared context provides the fundamental basis for meaningful communication. Thus, the information provided by an ASK system will correspond to the information needed by the user; the choices made available to him or her will be more relevant and easier to resolve; and the information conveyed in the interaction will be associated with the appropriate context in the user's memory.

The concrete realization of this principle in an ASK system depends upon a properly organized indexing network. Thus the construction of such a network must reflect a functional model of how the information is to be used. This model may derive from an abstract characterization of the task which the system is designed to teach, from the learning goals of the users of the system, or from the case-base of stories itself. Ideally, all three of these factors can be brought into alignment either by using stories told with an explicit pedagogical intent, or else by inducing an appropriate task orientation in students. In the absence of a task, the best the system can do is to implement a multiplicity of strategies, as for example is the case with the zooming system in ASK Michael. Armed with a task, the user of an ASK system—or indeed, of any hypermedia system—understands what he or she needs to learn; without a task, the user can only be guided only by local curiosity which cannot provide the context to sustain an interaction, and will incorporate new information, if at all, in ways that cannot be predicted.

Note

1. We have speculated on the pedagogical utility of providing a strong task focus by placing the student in the role of a President who must make difficult decisions and justify them in terms of current events and historical precedents. We believe that such an imposed task would be pedagogically useful, however the effectiveness of transfer from this particular task to one of historical analysis is an open question.

Acknowledgments: We thank Roger Schank and Chris Riesbeck for their help in the design of the ASK systems. ASK Tom was built with the assistance of Jorn Barger, Matt Greising, and Diane Schwartz. ASK The President was built with the assistance of Andy Fano, Bob Hooker, Mindy Wallis, and Jim Skinner. This research was supported in part by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058 and the Office of Naval Research under contract N00014-90-J-4117, by the Office of Naval Research under contract N00014-89-J-1987, and by the Air Force Office of Scientific Research under contract AFOSR-89-0493. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner, and from IBM.

References

Brown, J., Collins, A., and Duguid, P. (1989, January/February). Situated cognition and the culture of learning. *Educational Researcher* 32-42.

Conklin, E. (1987, September). Hypertext: An introduction and survey. *IEEE Computer* 2(9):17-41.

Porter, M. (1990). The competitive Advantage of Nations. New York: The Free Press.

Riesbeck, C., and Schank, R. (1989). (Eds.) *Inside case-based reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. (1982). *Dynamic memory: A theory of reminding and learning in computers and people*. Cambridge, England: Cambridge University Press.

Schank, R. (1991). *Case-based teaching: Four experiences in educational software design*. Technical report 7, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Schank, R., Ferguson, W., Birnbaum, L., Barger, J., and Greising, M. (1991). *ASK TOM: An experimental interface for video case libraries*. Technical report 10, Northwestern University, The Institute for the Learning Sciences, Evanston, IL.

Schank, R. (1977). Rules and topics in conversation. *Cognitive Science* 1:421-441.

Sussman, G. (1975). *A computer model of skill acquisition*. New York: American Elsevier.

Supporting Learning on Demand with Design Environments

Gerhard Fischer
Department of Computer Science and Institute of Cognitive Science
Campus Box 430, University of Colorado at Boulder, Boulder, CO 80303

Abstract: Information overload, high functionality systems, and the rapid change of our world have created new problems and new challenges for education and training. New instructional approaches are needed to circumvent the unsolvable problems of *coverage* and *obsolescence*.

Learning on demand is a promising approach for addressing these problems because: (1) It contextualizes learning by allowing it to be integrated into work rather than relegating it to a separate phase, (2) it lets learners see for themselves the usefulness of new knowledge for actual problem situations, thereby increasing the motivation for learning new things, and (3) it makes new information relevant to the task at hand, thereby leading to more informed decision making, better products, and improved performance.

We have developed a *conceptual framework* (including the support of situated cognition, and the integration of action and reflection), a *system* architecture (integrating domain-oriented construction kits, hypermedia systems, and case-based libraries), and *prototype systems* (e.g., for architectural design, for software design, etc.) in support of learning on demand.

Learning on Demand–Why

The major justification for learning on demand is based on the necessity for education to be regarded as a distributed lifelong process; a process of learning new material as it is needed, which is even more pronounced once formal education is completed and practical work situations are encountered. Success models of learning on demand have existed in human societies in situations where a learner could afford the luxury of a personal coach and critic. By having a human coach or critic, learners can obtain support for *their* problem solving activities.

Coverage and Obsolescence

Technologically oriented design fields are growing and changing at an alarming rate. Learning everything in advance (e.g., in high functionality systems [Draper 1984]) is impossible because there are too many things to learn. The rapidly changing nature of available design objects (especially in all fields related to information technology) poses the problem that our knowledge needs to be updated constantly. The large and growing discrepancy between the amount of potentially relevant knowledge and the amount users can know and remember makes support for learning on demand one of the most important activities.

New Cognitive Science Approaches Toward Learning and Training

Current cognitive theory (Resnick 1989) emphasizes that learning is a process of knowledge construction, not one of knowledge recording or absorption (Papert 1986). Successful learners elaborate and develop self-explanations that extend the information in texts or other instructional materials. Learning is knowledge dependent; people use current knowledge to construct new knowledge and to restructure existing knowledge (Kintsch 1988). Learning is highly tuned to the situation in which it takes place (Greeno 1989; Lave 1988; Suchman 1987; Winograd, Flores 1986). No amount of knowledge of principles suffices to account for or guarantee the success of action in real-world problem situations. This challenges many of the basic assumptions about learning general problem-solving skills in a decontextualized way. Contextual elaboration is needed to devise specific courses of action in order to go beyond general procedural prescriptions. Learning on demand is a promising approach for addressing these challenges.

Integrating Action and Reflection

The work process of many practitioners can be described by a *reflection-in-action* model (Schoen 1983). Action is governed by a nonreflective thought process and proceeds until it breaks down. A breakdown occurs when the practitioner realizes that nonreflective action has resulted in unanticipated consequences–either good or bad. In order to notice the breakdown, the "situation has to talk back" to the practitioner. Reflection is used to repair the breakdown, and then (nonreflective) situated action continues. Reflection-in-action takes place within the action present, i.e., the time period during which the decision to act has been made but the final decision about how to act has not. Mechanisms in support of learning on demand are required to take advantage of breakdown situations.

Learning on Demand-How

Our early work in support of learning on demand focused on two types of systems: active help systems and critiquing systems. *Active help systems* (Fischer, Lemke, Schwab 1985) volunteer information. They address the problem that demands cannot originate from users in situations where users are unaware that additional relevant functionality to their task at hand exists in the system. The system itself has to take the initiative. One of the major problems with these type of systems is to avoid having them become too intrusive. *Critiquing systems* (Fischer et al. 1990) present a reasoned opinion about a user's product or action. Critics use knowledge of design principles to detect and critique suboptimal solutions constructed by designers and increase the "back-talk" of the situation.

Design Environments

Design environments are *domain-oriented* to support human problem-domain communication (Fischer, Lemke 1988), and with the help of critics they *integrate* knowledge-based graphic construction kits (in support of action) with hypermedia systems (in support of reflection) (Fischer, McCall, Morch 1989). Design environments enable designers *to learn* *within the context of their work on real-world problems* (Fischer, Lemke, McCall 1990). With such systems, learning does not take place in a separate phase and in a separate place but is integrated with working and contextualized by situations. The systems let users construct solutions to design problems, advise them when they are getting into trouble, and then provide directly relevant information. A basic assumption behind them is that by letting users see for themselves the usefulness of new knowledge for actual problem situations, the *motivation* of users will be increased for learning new things.

JANUS (Fischer, McCall, Morch 1989) is a design environment that supports architectural design, specifically floor plan layout for kitchen design. JANUS consists of two subsystems: JANUS-CONSTRUCTION, a knowledge-based construction component, and JANUS-ARGUMENTATION, a hypermedia argumentation component. The critics (Fischer et al. 1990) in JANUS-CONSTRUCTION are capable of evaluating design decisions (as expressed in the construction situation) and providing feedback when design principles are violated. The system not only provides the primitive elements of the design domain but also has an understanding of principles of good design in that domain. JANUS supports the construction of an artifact either "from scratch," by combining the primitives from a palette, or by modifying an already constructed artifact from its catalog.

The knowledge-based critiquing mechanism in JANUS bridges the gap between construction and argumentation. The critics provide the designer with immediate entry into the exact place in the hypermedia network where the argumentation relevant to the current construction task lies. The system supports learning on demand by providing argumentative information for construction effectively, efficiently, and without requiring the designers to (1) realize they need information, (2) suspect that needed information is in the system, or (3) know how to retrieve it.

An Architecture for Systems in Support of Learning on Demand

Based on our work on JANUS and a number of other design environment building efforts (Lemke, Fischer 1990; Dews 1989), we have developed a multifaceted architecture in support of learning on demand (see Figure 1).

In addition to the components already mentioned, the architecture contains a *specification component* allowing designers to articulate partial specifications, and a *simulation component* supporting users by allowing them to engage in usage scenarios with the artifact being designed. In order to support learning on demand, the architecture is integrated by the following components: CONSTRUCTION ANALYZER, CATALOG EXPLORER and ARGUMENTATION ILLUSTRATOR.

A prerequisite of learning on demand is that the demand must be noticed by the learner; therefore, it is necessary for the *situation to talk back*. However, for learners who do not have extensive experience in the domain, the situation is often mute unless the learning environment has a component that speaks up and points out issues that the designer may otherwise not have considered. The *CONSTRUCTION ANALYZER* fulfills this role by operating as a critiquing system. The critiquing approach *empowers users* by supporting learning on demand. Users increase their knowledge and their independence by working with systems that do not do the work for them, but make the arrangements necessary for them to do it themselves.

The *CATALOG EXPLORER* (Fischer, Nakakoji 1991) links the specification and the construction with the catalog. By exploiting information contained in a partial specification and a partial construction, the system presents the examples in the catalog with respect to their relevance to the task at hand as articulated by the partial specification and construction.

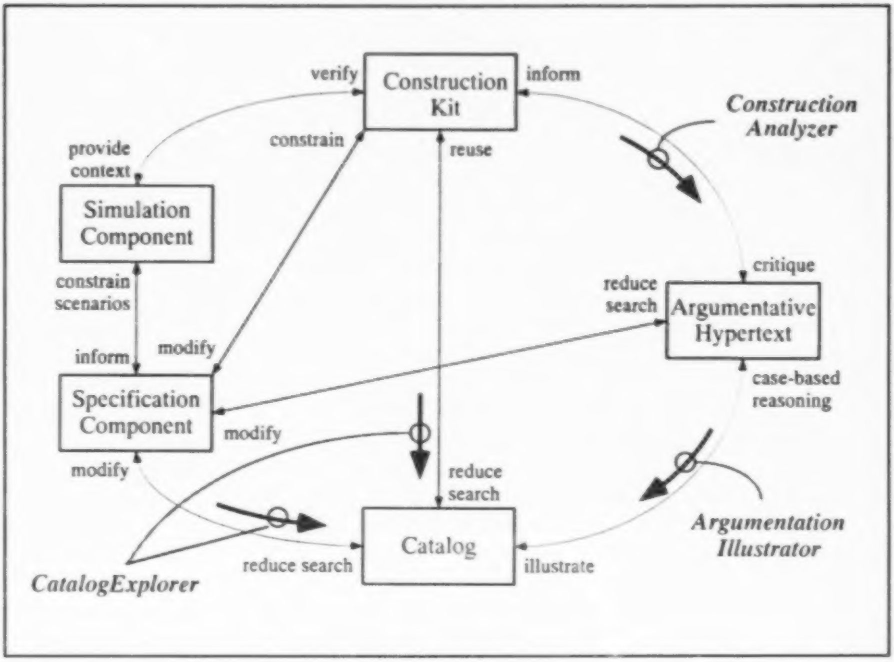

verify
Construction Kit
inform
Construction Analyzer
provide context
constrain
reuse
Simulation Component
constrain scenarios
critique
reduce search
Argumentative Hypertext
inform
modify
Specification Component
case-based reasoning
modify
modify
reduce search
Argumentation Illustrator
Catalog
reduce search
illustrate
CatalogExplorer

Figure 1. A Multifaceted architecture in support of learning on demand

The *ARGUMENTATION ILLUSTRATOR* is used to contextualize an argumentative principle in relation to a specific design. It provides a link back from argumentation to construction by making abstract principles concrete and ready to be integrated into the artifact under construction. Catalog examples support *case-based reasoning* (Riesbeck, Schank 1989), which complements generalized argumentative reasoning when principles are not sufficiently well defined. To support these learning processes, catalog examples must be annotated with design rationale that allow designers to assess the relevance of the example to the situation at hand. In addition to positive examples, the catalog also contains negative learning examples that show what design choices should be avoided and that assist learners in recognizing suboptimal designs.

Learning on Demand–Possibilities, Limitations, and Challenges

Beyond Tutoring and Open Learning Environments

The strength of tutoring systems (Wenger 1987) lies in their ability to teach basic concepts and skills of a problem domain. However, they cannot be designed with the concrete problem-solving situation of users in mind. Problems presented by tutoring systems are *prespecified* by the system, and therefore are not suited to support learning on demand.

Unlike tutoring systems, open learning environments (Papert 1980) do not suffer from the problem that presentation of instructional material is system controlled without regard to the learner's situation. But these environments have other shortcomings. Even the best environments do not cause or guarantee learning. In open learning environments, learners get trapped on suboptimal plateaus because they fail to discover the knowledge needed for better design, and there are no resources available to help learners detect mistakes or assist them when they get stuck.

Hypermedia Systems

Hypermedia is a promising new technology (Halasz 1988; Sculley 89; Weyer 1987) to be integrated into working and learning environments–but isolated hypermedia systems suffer from problems similar to those of open learning environments. The initial euphoria about hypermedia has been dampened by the realization that, unless carefully designed, hypermedia systems can fall short of realizing their potential (Fischer et al. 1988; Akscyn, McCracken, Yoder 1988). Finding information that is not directly linked to the current browsing position is difficult. Learners are often unfamiliar with the structuring principles of the information space and so have additional problems navigating to the desired piece of information. The richness of information may cause learners to get sidetracked, forget their original goals, study irrelevant information, and, as a result, "get lost in hyperspace." The integration of hypermedia systems into design environments eliminates some of these problems and makes complex information spaces accessible with a demand generated by the doing of a user.

Motivation

Users are willing and motivated to learn when the following conditions hold: (1) learning is actively desired by and controlled by them, (2) new information is easy to find they are successful in finding and using it, (3) users can see the immediate benefit of learning something new to their current working situation, and (4) their environments are intrinsically motivating allow users to achieve favorable results with a reasonably small amount of effort. The pain and the cost of acquiring new tools must be far less than the pain and the cost of trying to master difficult problems with inadequate tools. There is overwhelming evidence that self-directed (Brown, Palincsar 1989) and intentional learning (Bereiter, Scardamalia 1989) are some of the most important aspects of successful learning experiences.

Challenges for Psychology

Learning on demand raises many *psychological questions*. Our prototype design environments allow us to explore the following questions: (1) how does learning on demand compare with other, conventional forms of learning? (2) how can we characterize the class of users (e.g., novices, intermediates, experts) that benefit most from using an integrated design environment? (3) will designers with limited expertise understand the critics and learn from them to create reasonable designs? (4) what intervention strategies should the system use for displaying enough information at the right time without disrupting the work process?, and (5) when will designers suspend the construction process and access relevant information?

Challenges for System Building

System building efforts in support of learning on demand face the challenge of how a system can relinquish control of task selection yet maintain knowledge of users' goals, plans, and background knowledge. How can such systems be designed to function effectively in huge and potentially non-determinable solution spaces? Our work addresses this problem by modeling problem domains with design environments. Rather than representing solutions to individual problems, design environments support contextualized information access which has as its goal to deliver the right knowledge, in the context of a problem or service, at the right moment for a human professional to consider. A partial understanding of the task at hand (as expressed by a partial specification and a partial construction) allows the system to prioritize information spaces in support of learning on demand (Fischer, Nakakoji 1991).

Systems that allow learning to take place within the context of real problem-solving situations must avoid the "production paradox" (Carroll, Rosson 1987), where learning is inhibited by lack of time and working is inhibited by lack of knowledge. Learners must regard the time and effort invested in learning to be immediately worthwhile for the task at hand—not merely for some putative long-term gain.

Limitations of Learning on Demand

Beyond the possibilities that learning on demand can offer, one should not overlook a number of limitations: (1) the acquisition of certain essential skills should not be deferred until they are needed, because the time to learn them may be not available or the environment may be too dangerous for safe learning processes; (2) learning on demand is task driven and therefore may be limited to exposing users to isolated pieces of knowledge providing only limited support for learning essential principles; (3) users may encounter difficulties in decontextualizing knowledge so that it can be used in new settings, and (4) whereas learning on demand may be well suited for evolutionary extensions of a knowledge base, it may not support substantial restructuring because the additional features learned occur only in the neighborhood of what learners already know.

Conclusions

Learning on demand is necessary because learning in advance is no longer feasible in our high-technology world. More research needs to be done to determine whether learning on demand differs from other types of learning and, in particular, whether it is more or less effective. This will require a combination of psychological and computer science research, because learning on demand cannot be studied in isolation from systems supporting it, and support systems cannot be built without understanding the psychological mechanisms of learning on demand. Learning on demand may overcome some problems of schooling which alienate students from the world treating learning as an activity separate from the rest of life. Learning on demand is what truly sets computer-based environments apart from other media—and it may therefore be a unique research direction where technology can make a real difference in achieving new educational objectives and overcome basic limitations of our current educational approaches.

The research was supported in part by the National Science Foundation under grants No. IRI-8722792 and IRI-9015441, by the Army Research Institute under grant No. MDA903-86-C0143, and by grants from the Intelligent Systems Group at NYNEX and Software Research Associates.

Acknowledgements: The author would like to thank his colleagues (especially H. Eden, W. Kintsch, A. Lemke, C. Lewis, R. McCall, and P. Polson) and the members of the Human-Computer Communication group at the University of Colorado, who contributed to the development of the conceptual framework and the prototype systems discussed in this paper.

References

Akscyn, R.M., McCracken, D.L., Yoder, E.A. (1988). KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations, *Communications of the ACM,* Vol. 31, No. 7.

Bereiter, C. and Scardamalia, M. (1989). Intentional Learning as a Goal of Instruction, in L.B. Resnick (ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 361-392, ch. 12.

Brown, A.L. and Palincsar, A.S. (1989). Guided, Cooperative Learning and Individual Knowledge Acquisition, in L.B. Resnick (ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 393-451, ch. 13.

Carroll, J.M. and Rosson, M.B. (1987). Paradox of the Active User, in J.M. Carroll (ed.), *Interfacing Thought: Cognitive Aspects of Human-Computer Interaction*, The MIT Press, Cambridge, MA, pp. 80-111, ch. 5.

Dews, S. (1989). Developing an ITS in a Corporate Setting, *Proceedings of the 33rd Annual Meeting of the Human Factors Society* (Denver, CO), pp. 1339-1342.

Draper, S.W. (1984). The Nature of Expertise in UNIX, *Proceedings of INTERACT'84, IFIP Conference on Human-Computer Interaction,* Elsevier Science Publishers, Amsterdam, pp. 182-186.

Fischer, G., Weyer, S.A., Jones, W.P., Kay, A.C., Kintsch, W., and Trigg, R.H. (1988). A Critical Assessment of Hypertext Systems, *Human Factors in Computing Systems, CHI'88 Conference Proceedings* (Washington, DC), ACM, New York, pp. 223-227.

Fischer, G., Lemke, A.C., Mastaglio, T. and Morch, A. (1990). Using Critics to Empower Users, *Human Factors in Computing Systems, CHI'90 Conference Proceedings* (Seattle, WA), ACM, New York, pp. 337-347.

Fischer, G. and Lemke, A.C. (1988). Construction Kits and Design Environments: Steps Toward Human Problem-Domain Communication, *Human-Computer Interaction,* Vol. 3, No. 3, pp. 179-222.

Fischer, G. Lemke, A.C. and McCall, R. (1990). Towards a System Architecture Supporting Contextualized Learning, *Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence,* AAAI Press/The MIT Press, Cambridge, MA, pp. 420-425.

Fischer, G., Lemke, A.C. and Schwab, T. (1985). Knowledge-Based Help Systems, *Human Factors in Computing Systems, CHI'85 Conference Proceedings* (San Francisco, CA), ACM, New York, pp. 161-167.

Fischer, G., McCall, R. and Morch, A. (1989). JANUS: Integrating Hypertext with a Knowledge-Based Design Environment, *Proceedings of Hypertext'89* (Pittsburgh, PA), ACM, New York, pp. 105-117.

Fischer, G. and Nakakoji, K. (1991). Making Design Objects Relevant to the Task at Hand, *Proceedings of AAAI-91, Ninth National Conference on Artificial Intelligence,* AAAI Press/The MIT Press, Cambridge, MA, forthcoming.

Greeno, J.G. (1989). Situations, Mental Models, and Generative Knowledge, in D. Klahr, K. Kotovsky (eds.), *Complex Information Processing: The Impact of Herbert Simon*, Lawrence Erlbaum Associates, Hilldale, NJ, pp. 285-318, ch. 11.

Halasz, F.G. (1988). Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems, *Communications of the ACM,* Vol. 31, No. 7, pp. 836-852.

Kintsch, W. (1988). The Role of Knowledge in Discourse Comprehension: A Construction-Integration Model, *Psychological Review,* Vol. 95, pp. 163-182.

Lave, J. (1988). *Cognition in Practice*, Cambridge University Press, Cambridge, UK.

Lemke, A.C. and Fischer, G. (1990). A Cooperative Problem Solving System for User Interface Design, *Proceedings of AAAI-90, Eighth National Conference on Artificial Intelligence,* AAAI Press/The MIT Press, Cambridge, MA, pp. 479-484.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*, Basic Books, New York.

Papert, S. (1986). *Constructionism: A New Opportunity for Elementary Science Education*, Proposal to The National Science Foundation, MIT - The Media Laboratory, Cambridge, MA.

Resnick, L.B. (1989). Introduction, in L.B. Resnick (ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser*, Lawrence Erlbaum Associates, Hillsdale, NJ, pp. 1-24, ch. 1.

Riesbeck, C. and Schank, R.C. (1989). *Inside Case-Based Reasoning*, Lawrence [illegible]um Associates, Hillsdale, NJ.

Schoen, D.A. (1983). *The Reflective Practitioner: How Professionals Think in Action*, Basic Books, New York.

Sculley, J. (1989). The Relationship Between Business and Higher Education: A Prespective on the 21st Century, *Communications of the ACM,* Vol. 32, No. 9.

Suchman, L.A. (1987). *Plans and Situated Actions*, Cambridge University Press, Cambridge, UK.

Wenger, E. (1987). *Artificial Intelligence and Tutoring Systems*, Morgan Kaufmann Publishers, Los Altos, CA.

Weyer, S.A. (1987). As We May Learn, Multimedia in Education: Interfaces to Knowledge, *Education Advisory Council Conference Proceedings,* Apple Computer, pp. 87-103.

Winograd, T. and Flores, F. (1986). *Understanding Computers and Cognition: A New Foundation for Design*, Ablex Publishing Corporation, Norwood, NJ.

Towards Tutor Compilers: Self-Explanatory Simulations as an Enabling Technology

Kenneth D. Forbus
The Institute for the Learning Sciences, Northwestern University
1890 Maple Avenue, Evanston, Illinois 60201

Abstract: This paper discusses *self-explanatory simulations*, an integration of qualitative and quantitative techniques, which we believe will provide the foundation for a new generation of simulation-based tutoring systems. We describe the basic technology and decompose potential instructional uses along three dimensions, *type of explanations, artifact size*, and *processing locus*, indicating what progress seems needed for particular classes of applications.

1. Introductions

Simulation-based training provides a powerful technique for ITS. Examples include SOPHIE (Brown, Burton, and deKleer 1982), STEAMER (Hollan, Hutchins, and Weitzman 1984; Stevens et al. 1981) and RBT (Woolf et al. 1986). Their power stems from the ability of numerical simulation to provide a reasonably accurate depiction of artifact behavior and the use of direct-manipulation interfaces to make the simulation internals inspectable to some degree. But building simulation-based trainers can be very difficult, in part because building numerical simulations themselves is very difficult. In both STEAMER and RBT, pre-existing simulations were used as a starting point. Creating such simulations typically takes many person-years of effort. So while there are many potential applications for simulation-based training, the lack of usable simulators is a serious roadblock.

Even when simulators exist, they are often hard to use or unsuitable for training purposes. Just hooking up an existing simulator can be a herculean chore, requiring equal measures of archeology, operating-systems expertise, and luck. Most numerical simulators suffer the traditional problems of custom software: a lack of documentation and the existence of many implicit modeling assumptions. Intelligent tutoring and training systems should provide explanations, but grafting an explanation system on top of an existing numerical simulator is a difficult, and often impossible, task. In STEAMER, for instance, no qualitative explanations were provided in the main system and only a few modules used qualitative techniques. In part this was due to the primitive state of qualitative physics in the early 80's. But it was also due to the sheer difficulty of "spelunking" through the simulator to figure out just what parts of the steam plant were modeled, and to what degree of fidelity.

This paper explores *self-explanatory simulations* (Forbus and Falkenhainer 1990) as a way around this roadblock. Section 2 describes the basics of self-explanatory simulations. Section 3 examines this technology from the standpoint of building ITS's. We decompose potential systems along three dimensions: their explanation capabilities, the size of artifact involved, and the degree to which on-line processing is required. This decomposition is used to examine what may be practical in the near term versus what requires substantial additional basic research. Finally, Section 4 outlines some further issues and our plans to build a first-pass Tutor Compiler.

2. Self-Explanatory Simulations

The basic idea of self-explanatory simulations is to use the qualitative analysis of a system as a framework to organize a numerical simulation of that system. Consider the traditional *state space* formulation of dynamical systems. In this formulation a system has a set of *state variables* which suffice to completely determine its properties. The state variables for a spring-block oscillator, for instance, might be the position of the block and its velocity. These state variables may in turn determine other properties of the system (e.g., kinetic energy), but all other parameters are functions of these state variables. In Qualitative Process theory (Forbus 1984a), such state variables correspond to *directly influenced* quantities. In QP theory the directly influenced quantities cause in turn changes in other quantities, just as other parameters are determined by state variables in the state-space formulation. In a state-space the state of the system is described by a vector of values for the state parameters; while in qualitative terms it is described by a set of inequalities over the system's variables. Effectively, the qualitative description quantizes state space into regions of "equivalent" (i.e., qualitatively identical) behavior.

A simple system like a spring-block oscillator can be modelled by a single set of equations. But many systems are not so simple. For example, the set of equations relevant for modeling a can of water being heated on a stove is different from the set of equations that model what happens when that water is boiling. Traditional techniques focus on analyzing a given set of equations, and tend to ignore the modeling process itself. Qualitative physics provides tools for formalizing the modeling process, enabling the implicit assumptions of the modeler to be expressed and reasoned about. For example, a spring might behave linearly for small excursions but non-linearly for large displacements. Such changes in the relevant equations are then reflected in the quantization of state-space introduced by the qualitative description.

Already we see one role that qualitative analysis performs in simulation generation: constructing the relevant sets of equations. But there is more to it than that. First, suppose our simulation includes a notion of qualitative state as well as numerical state. This provides the potential for improved *self-monitoring*, i.e., detecting clashes between expectations expressed qualitatively and numerical predictions. Suppose, for instance, that one has modeled a spring-block oscillator in the usual way with the spring providing a restoring force. A naive user might try a spring constant with the wrong sign (corresponding to a "exacerbating force"), and the simulation will detect the inappropriate behavior which results and flag the relevant parameters for correction. Integrating the qualitative state with the numerical state also provides the basis for powerful explanation facilities, since the qualitative state can be used to access the set of equations describing the current behavior, the artifact's causal structure, and the qualitatively distinct behaviors which may occur next. These capabilities are illustrated in Forbus and Falkenhainer (1990).

This self-explanatory simulation concerns a large can containing a small amount of water sitting on a stove. In the dialogue below, the user has run the simulation and is viewing plots of the amounts of water and steam and their temperatures. Questions are posed in a formal query language, and paraphrased here in English. The answers were generated automatically by the simulator's explanation facility. Commentary is in italics.

Q: What is happening at t = 100.0 seconds?
A: Heat is flowing from STOVE to the water in CAN.

Qualitative explanations are available for every time tick in the simulation.

Q: What is the temperature of the steam?
A: Sorry, the temperature of the steam in CAN does not exist at 100.0 seconds.

The simulator knows that the numerical value it has for this tick is meaningless and does not report it.

Q: What happens next?
A: At 238.6 seconds the boiling temperature of WATER and the temperature of the water in CAN become equal. Then, heat is flowing from STOVE to the water in CAN and the water in CAN is boiling.

Here we see the qualitative and quantitative components interacting: The qualitative aspect identifies this point as interesting, the numerical aspect ascertains its exact time.

Q: What else might have happened?
A: The only other possibility was: The temperature of the water in CAN and the temperature of STOVE become equal, which leads to S2. In S2, nothing is happening.

The qualitative aspect of the simulation provides alternate possible behaviors.

Figure 1. An example of self-explanatory simulations

The causal structure imposed by the qualitative analysis is more than just a device for qualitative simulation and explanation. It plays a central role in building the numerical aspect of the simulator itself. For example, the order in which parameters are updated can be read directly off the graph of influences in the QP analysis of the system. Also, tests are incorporated to determine when the qualitative state changes, so that different code is used to reflect the different set of governing equations.

To summarize, a self-explanatory simulator combines qualitative and numerical representations to generate predictions and explanations of an artifact's behavior. The numerical component provides precise predictions of behavior, while the qualitative component explains this behavior in terms of the underlying qualitative model and set(s) of equations. (In this respect we differ from the IMTS approach (Towne and Munro 1988), which focuses on crafting simulations with just enough fidelity for providing on-screen animations and training students in troubleshooting.)

Self-explanatory simulators can be built automatically, assuming the following inputs: (1) a *domain theory*, expressed in QP theory, which describes the fundamental physical phenomena of interest in qualitative terms; (2) a *Math Model Library*, which provides fragments of equations for each set of qualitative proportionalities which may constrain a type of quantity in the domain theory; and (3) a *structural description* of the artifact or system to be modeled, including a set of modeling assumptions outlining the kinds of factors to be considered. Given (1) and (3), an *envisionment* is generated for the artifact, representing its quantized state space under the particular modeling assumptions. This envisionment is then analyzed in concert with the Math Model Library to produce a self-explanatory simulation.

3. A Design Space for Simulation-Based Tutors

There are many ways to build and use simulation-based tutors. To clarify the issues involved, we distinguish three dimensions which characterize potential applications. The location of the state of the art along these axes indicates what kinds of systems are (and are not) technologically feasible. We consider each dimension in turn, and then examine some specific potential applications.

Types of Explanations

Instruction can involve a range of explanations, from simple statements about a system's structure to causal explanations to analogies explaining a new phenomenon in terms of the familiar. While there are a host of issues concerning what should be explained (and how and when) which must be tackled by any ITS designer, our focus here is just on what kinds of information are in principle available.

Self-explanatory simulations provide several useful kinds of explanations directly. For any given time, one can ascertain (a) the equations governing the system, (b) the causal story linking the various system parameters, in terms of processes and influences, and (c) the numerical values of the system's continuous properties. Predictions can either be made via additional simulation, or via qualitative analysis. For example, if the state transitions of the envisionment used to generate the simulation are cached as part of the explanation system, alternate behaviors can be easily described (c.f. Figure 1). However, they provide no direct explanation of the teleology of the system or how the system might behave under different modeling assumptions than the simulation was generated for. Nor do they explain the fundamental principles of the domain itself in more detail, nor provide any sort of explanation that is not specific to the particular system the simulator was compiled for. To generate such explanations will require other facilities, built on top of the representations supplied by self-explanatory simulators.

Roughly, additional explanation facilities can be divided into two categories: *canned* and *generative*. Examples of canned explanation facilities include chunks of text describing the system's components, or hypermedia networks. Using hypermedia with self-explanatory simulations is a particularly interesting prospect. Self-explanatory simulations could become a new kind of node in the network, with the terms used in the explanation facility cross-indexed into the rest of the network.

Generative explanation facilities include for example the ability to recognize specific categories of behavior in functional terms. Determining whether or not an oscillation is damped, for instance, requires analyzing the behavior as it unfolds. Another useful generative capacity would be the ability to evaluate a student's analogies, suggesting corrections to avoid misconceptions and to lead to a deeper understanding. Obviously such capabilities would be very useful pedagogically, but currently these are matters of basic research.

Artifact Size

The operation of self-explanatory simulators can be made extremely fast. There is no reason to believe that they cannot asymptotically approach the speed of a traditional mathematical simulator, with clever design. Building them is another story. The current SIMGEN compiler relies on total envisionments as its starting point for generating behaviors. The size of an envisionment tends to be exponential in the size of the system modeled, which means that the current system is suitable only for small-scale examples, and could be applied to medium-scale systems only with substantial computing resources. We are currently exploring several techniques to solve this problem.

Processing Locus

Performing novel reasoning from first principles will always be slower than skill-based behavior. As Anderson (1986) demonstrated, a useful strategy for delivering instruction on affordable machines is to use substantial off-line resources to produce a skilled system that performs in a pre-selected subset of the domain. Self-explanatory simulators can be viewed as generalizing this technique to simulation-based systems. For some instructional tasks off-line reasoning and on-line simulation will suffice. However, there are other tasks which require on-line reasoning. For instance, evaluating a student's design requires the ability to rapidly analyze a new system.

What Can We Do When?

Now that we have some dimensions to characterize simulation-based trainers, we can examine some potential instructional applications for self-explanatory simulations and better understand what is needed to achieve them.

Active illustrations. Simulations which involve a handful of parts that are used to illustrate fundamental physical principles are perhaps the simplest instructional application. Explanations would be in terms of the QP domain theory, the artifact size is small, and the processing locus is totally off-line. Building such systems appears at this point to mainly require some software engineering work, to make the current version of SIMGEN more bulletproof and to make it produce better stand-alone runtime systems.

Subsystem trainers include systems with controls and possible faults, with roughly the complexity of an engineering watch station on a navy ship or an aircraft subsystem (e.g., an engine or hydraulic system). Building such systems will require extending self-explanatory simulations to use *action-augmented envisionments* (Forbus 1989). This extension seems straightforward, although the inclusion of fault models may require substantially increased compilation time.

Training simulators, such as STEAMER and RBT, involve a substantial (i.e., orders of magnitude) increase in complexity over subsystems trainers. Compiling self-explanatory systems on this scale will require a breakthrough.

Simulated laboratories and construction sets. Learning by building, where students create novel devices by combining stock parts, is pedagogically attractive (c.f. Forbus 1984b). However, they can require substantial on-line processing, since the structure of the artifact cannot be fixed in advance. Again, this will require a breakthrough.

Critics and coaches. Using trainers and construction sets often requires coaching, which ideally should be provided by the program itself. This requires expanding the kinds of knowledge beyond the physical principles encoded in a standard QP domain theory. For instance, analyzing modeling assumptions (Falkenhainer and Forbus 1991) is essential in understanding why perpetual motion machines don't work. Recognizing that a student's refrigerator design doesn't quite work correctly requires teleological reasoning. These issues require more basic research.

4. Discussion

Building ITS's is currently a labor-intensive enterprise, with thousands of hours invested for each hour of instruction delivered. Automation seems crucial. Ultimately, we want a full-fledged *Tutor Compiler*—a system which could produce ITS' given the description of the artifact, its intended users, and target hardware (Forbus 1988). We believe self-explanatory simulators are an important step towards that goal. Currently at ILS we are designing the Mark One Tutor Compiler, which will produce active illustrations and subsystem trainers which can run on small (IBM PS/2) computers. We hope to have this system operational sometime during fall of 1991.

One factor not mentioned in the design space for simulation-based tutors, but which is perhaps the most critical, is the quality and scope of the domain theory. Anderson (1988) points out that developing the domain theory is often about half of the total ITS-building effort. As we learn how to automate ITS construction more, we can only expect this fraction to grow. It will be a happy day when the biggest problem facing the builders of intelligent tutoring systems is teaching a tutor compiler what it should be teaching.

Acknowledgements: The work on self-explanatory simulations is carried out in collaboration with Brian Falkenhainer. The Mark One Tutor Compiler is being built in collaboration with Gregg Collins. Financial support has been provided by NASA Langley Research Center, Xerox, and IBM. The Institute for the Learning Sciences was established in 1989 with the support of Anderson Consulting, part of the Arthur Anderson Worldwide Organization. The Institute receives additional funding from Ameritech (an Institute Partner) and IBM.

References

Anderson, J.R. and Skwarecki, E. (1986). The automated tutoring of introductory programming. *CACM* 29(9):842-849.

Anderson, J.R. (1988). The expert module. In M.C. Polson and J.J. Richardson, *Foundations of intelligent tutoring systems*. New Jersey: Lawrence Erlbaum Associates.

Brown, J.S., Burton, R.R., and de Kleer, J. (1982). Pedagogical, natural language, and knowledge engineering techniques in SOPHIE I, II, and III. In D.H. Sleeman and J.S. Brown (Eds.), *Intelligent tutoring systems*. London: Academic Press.

Falkenhainer, B. and Forbus, K. (1991, Fall). Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, to appear.

Forbus, K. (1984a). Qualitative process theory. *Artificial Intelligence* 24.

Forbus, K. (1984b). An interactive laboratory for teaching control system concepts. BBN Tech Report No. 5511.

Forbus, K. (1988, Fall). Intelligent Computer-aided Engineering. *AI Magazine*.

Forbus, K.D. (1989). Introducing actions into qualitative simulation. *Proceedings of IJCAI-89*.

Forbus, K. and Falkenhainer, B. (1990). Self-Explanatory Simulations: An integration of qualitative and quantitative knowledge. *Proceedings of AAAI-90* Boston, MA.

Hollan, J., Hutchins, E., and Weitzman, L. (1984, Summer). STEAMER: An interactive inspectable simulation-based training system. *AI Magazine*.

Roberts, B. and Forbus, K. (1981). The STEAMER mathematical simulation. BBN Tech Report No. 4625.

Stevens, A., Roberts, B., Stead, L. Forbus, K., Steinberg, C., Smith, B. (1981). STEAMER: Advanced computer-aided instruction in propulsion engineering. BBN Tech report.

Towne, D.M. and Munro, A. (1988). The intelligent maintenance training system. In J. Psotka, D.L. Massey, and S.A. Mutter, *Intelligent tutoring systems lessons learned*. New Jersey: Lawrence Erlbaum Associates.

Woolf, B., Blegen, D., Jansen, J., and Verloop, A. (1986). Teaching a complex industrial process. *AAAI-86*.

Discourse-Based Second-Language Learning Environments

Carl H. Frederiksen and Janet Donin
Laboratory of Applied Cognitive Science, McGill University
3700 McTavish St., Montreal, Canada H3A 1Y2

Michel Decary
Recherches Neurox Inc.
1405 rue Bishop, Suite 100, Montreal, Canada H3G 2E4

Bruno Emond
Laboratory of Applied Cognitive Science, McGill University
3700 McTavish St., Montreal, Canada H3A 1Y2

Abstract: The acquisition of communicative competence in a second language requires that learners acquire facility in morpho-lexical, syntactic and semantic processing of sentences in the second language. It also requires that they integrate these sentence-level processes into their already developed meaning-oriented systems for processing connected discourse so that they can comprehend and produce discourse effectively in the second language. This paper reports progress in developing a performance environment for facilitating the acquisition of communicative competence in a second language that is based on current stratified models of discourse comprehension. The approach taken to instruction adopts a discourse communication framework in which the systematic study of syntactic forms is introduced into the performance environment in the form of an embedded tutor. This tutor is designed to develop expertise in the comprehension and production of morpho-syntactic expressions in the second language. To facilitate the integration of sentence-level processing skills into the learner's full discourse processing system, emphasis is placed on the semantic functions of morpho-lexical and syntactic structures in communicating propositional meanings, and on the comprehension and use of written discourse in authentic situations.

Second-language learning is similar to other learning situations in which what is acquired is competence in performing a class of complex information processing tasks. In second-language learning, the competence that is acquired is in the comprehension and production of connected discourse in a second language. What makes second-language learning unique are the particular representations and processes that are involved, and their overlap with those already developed in the first language. Representations shared with the first language include semantic representations of sentence meanings (i.e., propositions) and conceptual knowledge communicated by discourse (semantic networks and conceptual frames). Shared processes include operations in the generation and manipulation of semantic representations at the sentence and discourse levels. To acquire discourse competence in a second language, learners must acquire component processes for morpho-lexical, syntactic, and semantic processing at the sentence level that are specific to the language being learned, and they must integrate these processes into their already developed meaning-oriented systems for discourse comprehension and production.

Second-language instruction traditionally has emphasized the teaching of vocabulary, morphology and syntax, but in recent years has shifted its emphasis to the development of discourse communication skills. The assumption in this more recent approach has been that syntactic competence in the second language will be acquired naturally within the communicative situation. However, it is not obvious that such incidental learning of morpho-syntactic rules in a second language would be efficient or complete. Nor is it clear how systematic instruction in morpho-syntactic processing could be integrated into communicative tasks promoting the development of discourse processing skill in the second language. Computer applications in the field of second-language tutoring usually have consisted of grammar, spelling or vocabulary tutors, and educational computer applications usually have not been designed to teach discourse communication skills in a second language.

This paper reports progress in developing a performance and tutorial environment for acquiring discourse comprehension competence in a second language that is based on current stratified models of discourse comprehension. It is intended to operate within a discourse communication framework, but to reintroduce the systematic study of syntactic forms into the communicative environment. However, it differs from earlier approaches in that the acquisition of morpho-syntactic rules is motivated by the functions of morpho-syntactic forms in communicating meaning.

Discourse Processing and Second-Language Learning

Efforts to model the cognitive processes and representations involved in discourse comprehension have resulted in a *stratified model* of discourse processing in which representations are generated for discourse at multiple levels (Frederiksen and Donin, 1991). The levels of representation identified by this model include: (a) representations of the morpho-lexical and syntactic structures of natural-language expressions, (b) semantic representations of the propositional meanings expressed in linguistic structures, and (c) frame-level conceptual representations of declarative knowledge or situations represented in connected discourse. The stratified model provides a framework for identifying quasi-independent component processes in text comprehension and production, and for modelling their interactions in terms of a modular multilevel system. In this model, processing at one level is viewed as operating on the outcomes of processing at other levels. However, processes may be operating in parallel with individual processes operating whenever they have data. For example, in comprehending a discourse in a particular situation, a reader or listener constructs a global knowledge representation from propositions derived from lexical and syntactic information. At the same time, knowledge representations that the reader or listener already possesses may be used both in the construction of a text representation and in processing at lower levels. The extent to which component processes are modular and the manner in which they interact with or influence one another are important areas of current research.

The stratified model of discourse processing has important consequences for second-language learning. First, it recognizes that second-language learners must acquire automatized component processes for morpho-lexical and syntactic processing at the sentence level that are specific to the language being learned. However, it also makes explicit that the learning of rules that map between morpho-syntactic structures and semantic representations at the propositional level is a major aspect of second-language acquisition. Second, the stratified model recognizes that these language-specific component processes must be integrated into the complete meaning-oriented discourse comprehension (or production) system. Therefore a tutor should develop expertise in the component processes specific to a second language in such a way as to facilitate their integration into a learner's full discourse processing system by emphasizing in instruction the *semantic functions* of morpho-lexical and syntactic structures in communicating propositional meanings.

Adult second-language learners already know how to generate a variety of well-formed expressions to express a given propositional meaning in their first language. They also know how to interpret these semantically-equivalent expressions in terms of a common meaning representation. What they must learn is how to generate the set of alternative well-formed expressions that may be used to express a given propositional meaning in a second language and how to interpret the propositional meaning of such expressions. Once students learn syntactic encoding and semantic-interpretation rules and acquire skill in applying them in sentence generation and comprehension tasks, these skills must be integrated into a well-functioning discourse comprehension and production system extended to their second language.

The design of an intelligent tutoring system to build these skills will require not only a specification of morphological and syntactic rules for the second language, but also a well-defined semantic representation for propositions and an explicit specification of rules relating propositions to morpho-lexical and syntactic structures. Finally, a tutoring system should be a part of a performance environment in which learners gain experience in comprehension and production tasks in the second language.

CODA-X: A Computational Model of the Semantic Interpretation of Sentences

We have been developing a computer system for the semantic interpretation of English called CODA-X that generates well-formed representations in a propositional meaning-representation language from syntactically analyzed natural language inputs (Frederiksen and Décary, 1990; Frederiksen, Décary, and Emond, 1990). In CODA-X, propositions are expressions in a meaning-representation language that is defined by rules in a recursive context-free semantic grammar (Frederiksen, 1985). Attached to all semantic production rules in the grammar are specifications of sets of syntactic patterns that can be used to express them in a specific language. Sentences are analyzed by a semantic parser that applies rules in the semantic grammar in a top-down analysis (with backtracking) of syntactic parse trees generated previously by a syntactic parser. Semantic parse trees (i.e, propositions) are generated as syntactic patterns and lexical information are found (in the syntactic parse tree for the input sentence) that are consistent with the application of particular semantic rules. An example of a semantic parse tree is given in Figure 1.

The approach taken to semantic analysis in CODA-X has a number of characteristics that make it pertinent as an expert domain-knowledge model for a second language tutoring system. First, CODA-X contains an explicit semantic grammar defining well-formed expressions (propositions) in a meaning-representation language (MRL) (Guenthner, Lehnmann, and Schönfeld, 1986; Jackendoff, 1983; Kamp, 1981). Second, sentence meanings are represented in terms of semantic parse trees. Propositions are assigned a structure by application of rules in a context-free semantic grammar. Third, representations

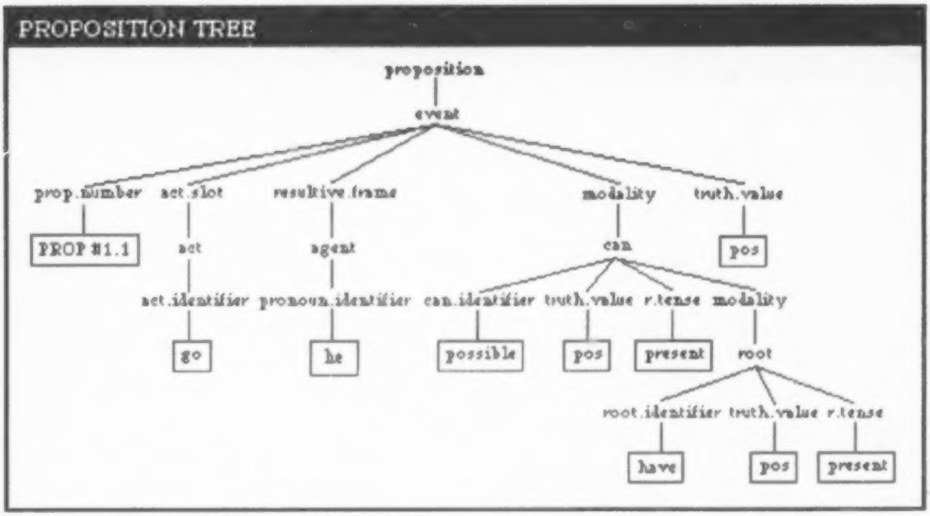

PROPOSITION TREE
proposition
event
prop.number
act.slot
resultive.frame
modality
truth.value
PROP #1.1
act
agent
can
pos
act.identifier
pronoun.identifier
can.identifier
truth.value
r.tense
modality
go
he
possible
pos
present
root
root.identifier
truth.value
r.tense
have
pos
present

Figure 1. Proposition tree for the sentence *it is possible that he has to go*

in the MRL employ explicitly defined relational categories identified in research on natural language semantics (e.g., Jackendoff, 1983). Fourth, meaning representations generated during semantic interpretation (semantic parse trees) may be transformed into equivalent conceptual graphs (i.e., semantic networks) in which all relations are specified by the grammar (as relational categories) and have a well-defined meaning (Sowa, 1984). These may also be expressed in predicate-argument form and are equivalent to expressions in a first (or higher-order) predicate calculus. Finally, input to the semantic interpreter is a syntactic parse tree for which a meaning representation is generated (Woods, 1978).

These characteristics of CODA-X are consistent with modular and stratified models of human discourse processing and are the main mechanism for relating semantic representations to linguistic structures. CODA-X combines these characteristics into a coherent system for semantic interpretation. The implementation of this approach to semantic analysis led to a distinction between "static" and "dynamic" rules. The "static" and "dynamic" rules represent respectively: (a) the recursive context-free grammar for propositions, and (b) the set of parameters, conditions and actions associated with each "static" production rule in the grammar that identify linguistic structures associated with the rule. This distinction means that the user has access to both a context-free formulation of the semantic grammar (the static rules) as well as the specific conditions of application of static production rules (the dynamic rules). The distinction between "static" and "dynamic" rules also supports the generality of the system because it allows the user to have access to: (a) the grammar determining the well-formedness of expressions of the meaning-representation language (independently of its application to any particular natural language), and (b) the specific conditions required to apply this grammar to a specific natural language. The static rules of the grammar define the propositional representations of sentence meanings, and the dynamic rules define the mapping from natural language expressions to propositions.

In CODA-X, static and dynamic rules are essentially declarative information. However, the dynamic rules have a procedural interpretation according to a well-defined inference engine that works on those rules. Although the CODA-X knowledge-base was developed to implement a semantic interpreter for natural language, its declarative representation of rules relating semantic representations to syntactic patterns (as packets of dynamic rules associated with static production rules in the proposition grammar) makes it potentially suitable as an expert knowledge-base for tutoring systems for learning the linguistic structures that express propositional meanings. The specific nature of static and dynamic rules will be further described in discussing the use of the CODA-X grammar as an expert knowledge-base for second-language tutoring systems. For a more complete description of the CODA-X grammar, dynamic rules and inference engine, see Frederiksen and Dècary (1990).

A Tutoring System for Acquiring Linguistic Knowledge

How can the CODA-X rule base can be used to organize an environment for learning the syntax-semantics mappings of English? Some of the possibilities may be seen by considering the example of the modal system. Suppose that we are considering a component of a tutoring system that is teaching a user how to understand and produce expressions in a second language that express modal information. A typical sequence of events that might take place in a tutorial environment in which a person was learning the modal system of English might be as follows.

First, one would begin by learning the semantics of modality with reference to examples in the learner's first language. Users would be given examples that illustrate different types of situations of use of different types of simple modal and complex modal expressions. These could be given by sentences or discourse in the users' first language. The series of examples and their explanations would be generated from the static rules for modality. For example, an event could be repeatedly presented with type of modality varied. The explanations would highlight the semantics of modality. A pedagogically appropriate representation based on the semantic parse tree for the examples would be presented and used as a basis for explaining the semantic rules. The proposition tree could be displayed as a parse tree, a simplified network, as a predicate-argument structure, or lists of properties. A student would learn to answer questions about the meaning of modal expressions (e.g., type of modal), and explanations would be given to explain categories, structures of complex modals, or answers to questions.

Once users had learned a particular semantic structure, they could request paraphrases of the original sentence. The system would generate alternative expressions that could be used to represent the proposition (e.g., the particular type of modal) as specified by the dynamic rules. For each alternative expression, an explanation of the particular syntactic structure would be given. Then the user would begin to explore the expressive possibilities of the second language. For example, the user could begin by identifying lexical equivalents for words in the semantic representation (with the help of a dictionary if necessary). Then the system would generate sentence alternatives expressing the proposition in the second language. Explanations of applicable grammatical rules of the second language would be available to the user. The user would practice comprehending examples of expressions of modality in the second language, with appropriate tests of comprehension. At a later stage, the user would be given sentences in the second language and asked to generate sentences in the second language to express variations in modality. The system would carry out syntactic and semantic analysis of the user's responses, and identify the user's errors keeping track of what rules the student had mastered. By this means, the student would acquire skill in comprehending and producing modal expressions in the second language. The motivation for the instruction would always be to learn how particular types of meanings are expressed in the second language. The training would capitalize on learners' knowledge and expertise in their first language to facilitate and motivate learning of the second language.

A Performance Environment for Developing Proficiency in Discourse Processing

As learners acquire facility in comprehending and expressing propositional meanings in their second language, they must successfully embed these procedures in their general discourse-processing system. Computer environments for second-language learning must facilitate the development of proficiency in discourse comprehension and production in the second language and maintain a focus on the main purpose of second-language learning which is to acquire communicative competence in the second language. Therefore, it is our view that a second-language learning system must be built around an authentic performance environment in which learners use and produce discourse in the second language in realistic situations that reflect their natural contexts of discourse communication. For example, nurses should be able to obtain practice in the use of discourse in the second language that is typical of the discourse and communicative situations that they encounter in their normal work settings. It will be important to design discourse and communicative situations that are both authentic and that permit practice with discourse that incorporates linguistic structures that have been studied in the tutorial environment.

For example, exercises involving comprehension of texts in the second language can incorporate sentence structures and semantic structures that have been studied. By interspersing these structures throughout texts, users can learn to integrate newly acquired semantic and syntactic processing skills in the second language into their conceptual discourse-processing system. By this means, users would learn to apply their knowledge of semantic and syntactic rules, of alternative syntactic structures that can be used to express propositional meanings (i.e., the dynamic rules), skill in producing alternative sentence structures to express meaning, and proficiency in comprehending a variety of sentence forms.

The kind of performance environment we are designing centers around connected discourse and its situations of use. A discourse should not be regarded as a static entity; rather it should be used to provide dynamic access to a variety of information, exercises and forms of expression. For example, a sentence can provide access to a tutor, paraphrases, exercises requiring paraphrasing or modifying its meaning, or comprehension questions. Students can be evaluated in terms of their performance on a variety of tasks reflecting different component skills in comprehension and production of discourse in the second language, and a variety of pedagogical strategies can be used to build discourse-processing expertise in the second language. The tutorial system for building syntactic and semantic processing competence at the sentence level can be made available as a help system in this performance environment, and the extent of system support for comprehension or production processes can be gradually reduced as the student becomes more proficient. We like to think about this second-language performance environment as simulating an instructional situation in which students comprehend or produce discourse with an "expert language teacher" always available to assist them by explaining different points of grammar, demonstrating alternative expressive possibilities, and assisting in comprehension and production. In this way, systematic instruction can be embedded in natural performance situations and made pertinent to everyday problems of discourse communication in the second language.

This research was supported by grants from the Ministere de l'Enseignement superieur et de la science, Quebec, and from the Social Sciences and Humanities Research Council of Canada. The figure is reproduced from Frederiksen, C.H., Donin, J., Decary, M., Emond, B. & Hoover, M. Semantic Discourse Processing and Tutorial Systems for Second-language Learning. In M.L. Swartz & M. Yazdani (Eds.), The Bridge to International Communication: Intelligent Tutoring Systems for Foreign Language Learning. *Heidelberg: Springer-Verlag, in press. The reader is referred to this chapter for a more detailed discussion of the ideas presented here.*

References

Frederiksen, C.H. (1986). Cognitive models and discourse analysis. In C.R. Cooper and S. Greenbaum (Eds.), *Written communication annual: An international survey of research and theory, vol. 1: Studying writing: Linguistic approaches.* Beverly Hills, CA: Sage.

Frederiksen, C.H., and Donin, J. (1991). Constructing multiple semantic representations in comprehending and producing discourse. In G. Denhière and J.-P. Rossi (Eds.), *Texts and text processing.* Amsterdam: North Holland.

Frederiksen, C.H., and Décary, M. (1990). *Cognitive discourse analysis Xerox 1186, Version 1.0.* Technical Manual, Laboratory of Applied Cognitive Science, McGill University, Montrèal, Canada.

Frederiksen, C.H., Décary, M., and Emond, B. (1990) *CODA-X: A semantic parsing approach to natural language interpretation.* Technical Report, Laboratory of Applied Cognitive Science, McGill University, Montrèal, Canada.

Guenthner, F., Lehnmann, H., and Schönfeld, W. (1986). A theory for the representation of knowledge. *IBM Journal of Research and Development, 30(1)*:39-56.

Jackendoff, R. (1983). *Semantics and cognition.* Cambridge, MA: MIT Press.

Kamp, H. (1981). A theory of truth and semantic representation. In J.A.G. Groenendijk, T.M.V. Janssen, and M.B.J. Stokhof (Ed.), *Formal methods in the study of language.* Amsterdam: North Holland.

Simmons, R.F., Burger, J.F., and Schwarcz, R.M. (1968). A computational model of verbal understanding. *AFIPS*, 441-456.

Sowa, J.F. (1984). *Conceptual structures: Information processes in mind and machine.* Reading, MA: Addison-Wesley.

Woods, W.A. (1978). Semantics and quantification in natural language question answering. In M. Yovits (Ed.), *Advances in computers, vol. 17.* New York: Academic Press.

Cognitive Strategies, Learning, and Educational Software

Douglas Frye
Computer Science Department
Yale University, 51 Prospect Street, New Haven, CT, U.S.A. 06520

Abstract: Students employ active cognitive strategies in learning. Designing educational software to support cognitive strategies represents a different approach to instruction. To test this new approach, we implemented a set of strategy-based computer programs (KIDSCOUNT) to teach elementary school children early mathematics. The software proved to be highly effective. Tests of the programs with public school children show that they produce significantly better math learning, and that the emphasis on strategies is responsible for the students' improvement. Specifically, in the present experiment, it was found that giving students computer instruction in the strategy of *counting all* improved their addition performance, the improvement generalized from the computer to problems presented on paper, and the strategies the students used changed as a consequence of the instruction. A highly similar version of the computer software that did not emphasize strategies produced no gain over classroom controls. The value of strategy-based instruction is that it furnishes students with effective means of learning rather than only focusing on what is to be learned.

Cognitive Strategies, Learning, and Educational Software

From at least the time of Piaget, students have been recognized as active learners. They rely on strategies to acquire knowledge. Cognitive strategies make learning more efficient. They may even be necessary for learning to occur. The adoption of new strategies is an essential feature of increasing expertise and development.

If strategies are important in learning, then instruction that emphasized strategies might well improve learning. Strategies have not typically been the focus of instructional computer systems. Both traditional computer-assisted instruction (CAI) and intelligent tutoring systems (ITSs) have tended to concentrate on knowledge rather than strategies. Although they do so in different ways, both use student errors, or gaps in knowledge, to guide instruction. Sleeman (1989) has recently questioned whether this assumption is necessary. He has found that reteaching students what they are trying to learn may be as effective as adapting instruction according to the specific errors students make.

Organizing instruction around strategies requires teaching specific strategies that aid learning without correcting each misconception a student encounters while learning. We have begun to investigate the strategy-based approach by designing a computer instructional system, KIDSCOUNT, for teaching elementary school children basic arithmetic—number, counting, cardinality, addition, and subtraction—through strategies. We have carried out a series of studies with the system to evaluate its effectiveness. The research also examines the more general questions of whether exposing students to instruction in strategies encourages strategies use and whether specific strategies are associated with improved learning.

Early arithmetic was chosen for the project for two reasons. The first was that specific strategies in early arithmetic have been identified and extensively researched (please see Ginsburg, 1983; Carpenter, Moser & Romberg, 1982; or Fuson, 1988 for reviews). It is even known that these strategies follow a rough developmental progression (Siegler & Jenkins, 1989; Steffe & Cobb, 1988). So, for example, when children first learn to add, they typically employ *counting all* (4 + 3 is solved by counting 1,2,3,4,5,6,7). Later, they are more likely to adopt a strategy of *counting on* (4 + 3 is solved by counting 4,5,6,7). Later still, they are able to retrieve the answer for simple problems from memory. Because the strategies in early mathematics are known, it is feasible to design instructional sequences for them.

Early math was also chosen because it is a weak point for many children in this country. Poor performance in math is evident in surveys of American students alone (Dorsey, Mullis, Lindquist & Chambers, 1988), and in comparisons of American children with those in China, Japan, and other industrialized countries (Uttal, Lummis, & Stevenson, 1988). The comparisons to other countries show that American children begin to fall behind as early as the first year of school (Stevenson, Lee, & Stigler, 1986). It may be that American children have more difficulties with mathematics because many schools emphasize rote exercises in math rather than problem solving strategies (Dorsey et al., 1988).

Our studies of the KIDSCOUNT system have all followed a similar design. Two groups of students are given experience with the computer except that in one case the software emphasizes strategies and in the other it does not. The strategy computer and nonstrategy computer groups are both compared to a classroom control group that does not use the computer. This design makes it possible to determine if improvement in the strategy computer group is only due to extra instruction on the computer or if the emphasis on strategies is responsible. Children in all three groups are given pre- and posttests on a variety of math tasks off the computer to test if any improvement in learning generalizes to problems given on paper. The strategies that each of the children use on the tests, and while working with the computer, are observed so that it is possible to discover how they change with the different forms of instruction.

Brief Description of the KIDSCOUNT Software

The software currently includes twelve programs for teaching number, counting, cardinality, addition, and subtraction. Each of the programs concentrates on one basic strategy in arithmetic. The current version of the system is implemented in Common Lisp in an object-oriented style. The software runs on personal computers that have been equipped with touchscreens. We have found that young children can more easily operate touchscreens than the mouse (Lu, Bloom & Frye, unpublished) or keyboard. The system relies on color graphics, basic animation, and realistic voice to convey arithmetic strategies to the student. It presents spoken numbers, written numbers, graphical objects, and sets of objects that can be joined and broken apart. The computer has two advantages in presenting strategies in that it can endow objects or "manipulatives" on the screen with a wide variety of different behaviors (e.g, sound, labels, movement), and can link different representations on the screen so that changes in one are reflected in the others (e.g., change in number of objects in a set is automatically reflected in number in an arithmetic equation).

Example Study from Addition

The study presented below examined the effects of providing kindergarten and first grade children with experience with a specific addition strategy on the computer. The study is one in a series that includes others on number learning and subtraction. To date, they have all produced a similar pattern of results. The main goal of all of the studies is to determine if giving the children access to a computer environment that supports appropriate counting strategies can improve their grasp of early mathematics.

Design. The addition study, like the others in this series, followed a between group design with three groups. One group of students was given experience on the computer with the counting all strategy addition program (strategy computer group—see below for details). Another group was given experience with the rote or noncounting-strategy addition program (nonstrategy computer group). The final group was not given an opportunity to use the computer (classroom control). Students were randomly assigned to the groups. All of the groups had the same classroom instruction.

Students. The students in the study were kindergartners and first graders drawn from four classrooms in two public schools in New Haven.

Data Collected. All of the students were individually tested before and after the study. The pre- and posttests were made up of tasks that examined counting (rote counting to highest number, counting backwards, counting starting at an arbitrary number, counting objects in linear and scattered spatial arrangements), addition (problems on paper with and without objects present, problems on the computer with and without manipulatives present, multiple addends on paper), and subtraction (problems on paper with and without objects). On all of these tasks, not only were the students' answers taken, but *how* they did the task was recorded, especially in regard to the strategies used on the addition and subtraction problems. If it was not obvious how the child solved a particular problem, they were asked to describe what they did.

In the two computer groups, a trained observer kept track how the students attempted to solve the addition problems when they were working with the computer. Every child's strategy on every problem was recorded. Again, if the strategy was not obvious, the student was asked to describe what he or she had done. The computer also recorded the objects each child touched on the screen and the elapsed times between touches throughout the study.

Instruction. The strategy computer group in the addition study were always given experience with counting all. After the student gave an answer to the problem, the computer also solved it by moving the manipulatives one by one across the screen and counting from 1 to the sum (e.g., for 4 + 3, "1, 2, 3, 4, 5, 6, 7") as shown in Figure 1. For the nonstrategy computer group, the manipulatives were moved as whole stacks corresponding to the addends and the computer spoke the problem and answer (e.g., "4 and 3 is 7").

In both computer groups, the children used the computer individually for approximately 15 to 20 minutes per week. During each weekly session they tried to solve 10 simple addition problems. The problems increased in difficulty over the course of the study. At the beginning, the problems had sums that were typically less than 5. They were between 15 and 20 at the end. There were 9 weekly sessions in total. *Fading* was introduced after the fifth session such that in half the problems presented the stacks of manipulatives were shown as single squares.

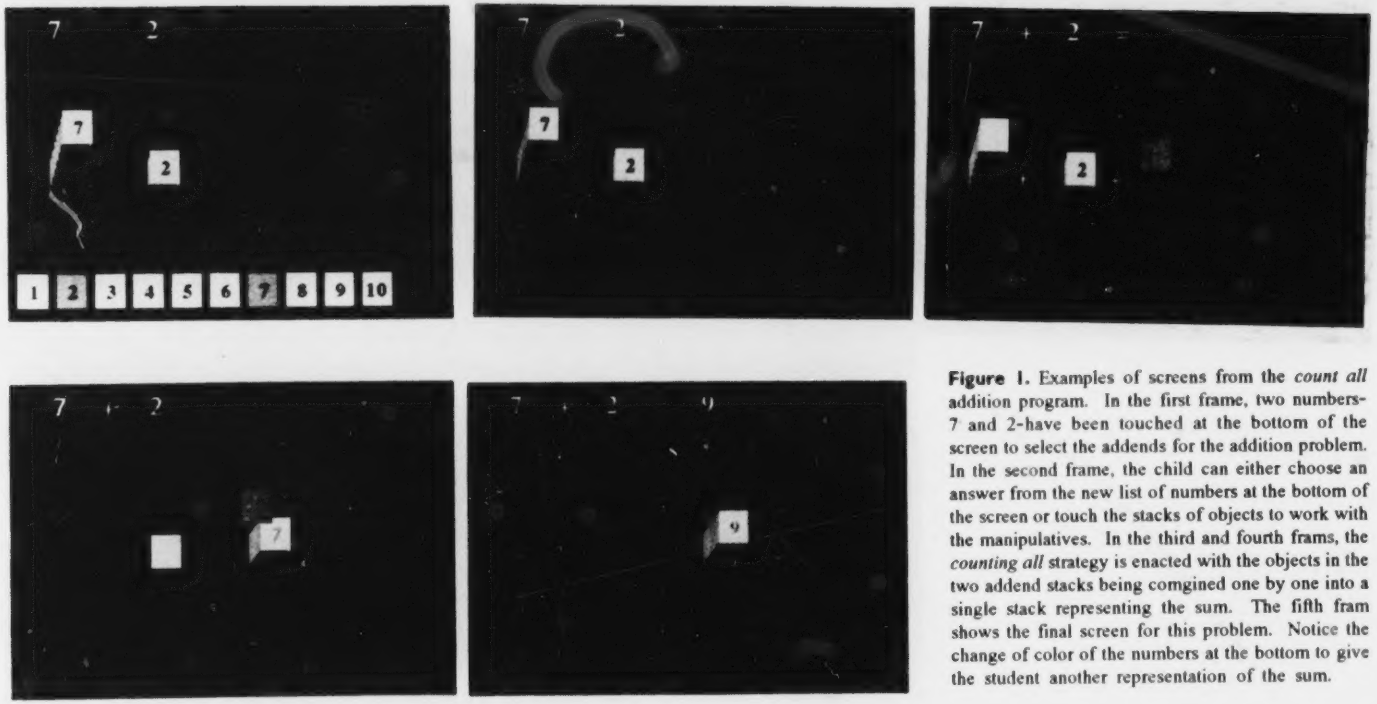


Figure 1. Examples of screens from the *count all* addition program. In the first frame, two numbers-7 and 2-have been touched at the bottom of the screen to select the addends for the addition problem. In the second frame, the child can either choose an answer from the new list of numbers at the bottom of the screen or touch the stacks of objects to work with the manipulatives. In the third and fourth frams, the *counting all* strategy is enacted with the objects in the two addend stacks being comgined one by one into a single stack representing the sum. The fifth fram shows the final screen for this problem. Notice the change of color of the numbers at the bottom to give the student another representation of the sum.

BEST COPY AVAILABLE

All of the children received similar forms of regular classroom instruction. The kindergartners had occasional activities with real manipulatives in the course of their classroom instruction. They were given group instruction in counting and very simple addition problems. The first graders were given extensive instruction in addition including practicing problems presented on the board and on worksheets.

Group Results. Figure 2 displays the pretest, training and posttest results for the three different groups in the addition study. The pretest reveals that the children on average could initially solve only about a third of the addition problems they were given. The problems were typically ones that had very small addends e.g., 2 + 3. Over the course of the nine weekly training sessions, the strategy computer group doubled their correct performance. From the second session to the ninth, they were significantly better than the nonstrategy computer group. In the posttest, the strategy computer performed significantly better ($F(2,66) = 6.96$, $p < .01$) than the nonstrategy computer and classroom controls groups, which did not differ from each other. Transfer was tested by administering posttest addition and subtraction problems on paper. The strategy computer group's advantage transferred to the addition problems on paper.

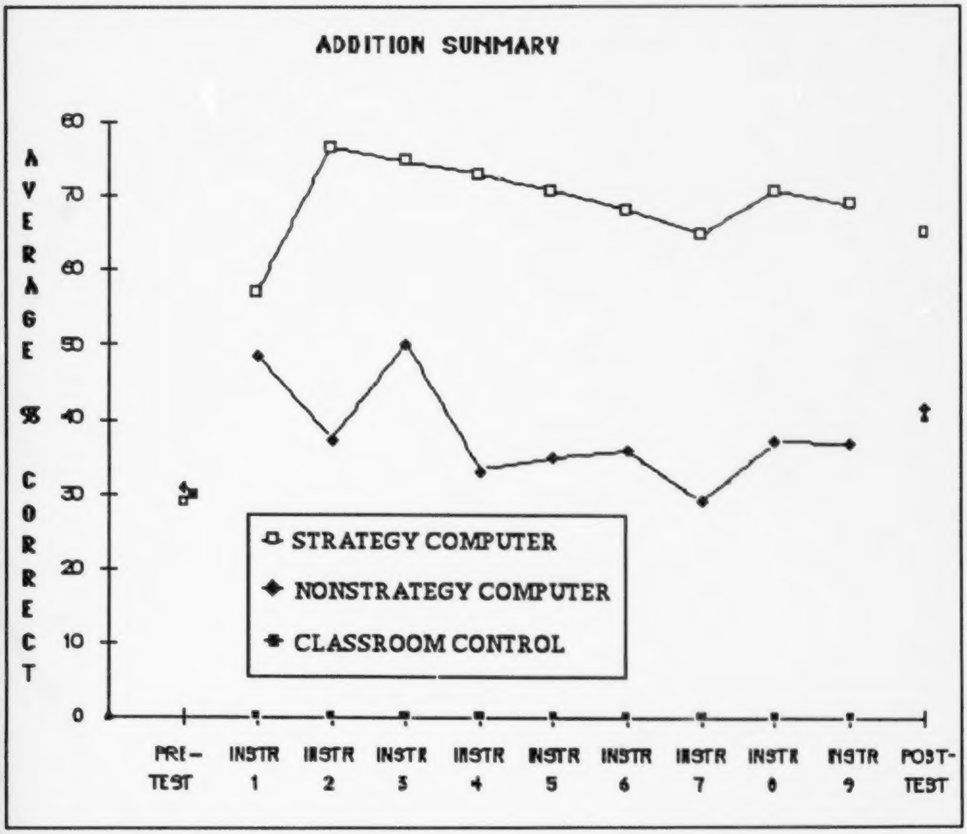


Figure 2. Percentage correct addition performance for the strategy computer group, nonstrategy computer group, and classroom control in the pretest, training and posttest phases of the addition study.

Strategies. The different groups employed different strategies in the posttest. The computer strategy group guessed less, and were able to retrieve from memory and count on more than the other two groups. The differences in strategy use were responsible for the better performance of the computer strategy group. When the children relied on memory, counting on or counting all, they were correct 95%, 83% and 76% of the time, respectively. However, when they had to guess, they were only correct 8% of the time.

Conclusion

The results of the experiment demonstrate that experience with strategies on the computer improves children's ability to solve simple addition problems. They indicate that this effect is not simply due to experience with the computer (or onscreen manipulatives, or more math practice, or rote learning) because the nonstrategy computer group did no better than the classroom controls even after nine weeks of instruction. The transfer results show that what the students learned was not restricted to the computer. The improvement in the strategy computer group carried over to problems presented on paper, thereby representing a genuine advance. Experience with the computer changed the addition strategies of the computer strategy group, which in turn improved their performance on the posttest. There is evidence that the instruction caused the computer strategy students to progress in the sequence of addition strategies given that they adopted strategies that were more advanced than they were trained on.

The positive findings in the study, and in others in this series, suggest that strategy-based instruction can promote effective learning with computers. The negative findings are also of interest. The nonstrategy version of the software, which more of less required the students to learn by rote, had virtually no effect. The difference confirms that strategies are an integral part of learning. Studies of the other specific strategies students employ could offer new opportunities for understanding how they learn.

This research was supported by a grant from the Smith-Richardson Foundation.

Acknowledgements: Chengdong Lu served as the main programmer of the KIDSCOUNT software. Jodie Ambrosino and Funke Ewedemi contributed significantly to the design and conduct of the research.

References

Dorsey, J. A., Mullis, I. V. S., Lindquist, M. M., & Chambers, D. L. (1988). *The mathematics report card: Are we measuring up?* NAEP Report No. 17-M-01. Princeton, NJ: Educational Testing Service.

Carpenter, T., Moser, J. M., & Romberg, T. A. (1982). *Addition and subtraction: A cognitive perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Ginsburg, H. (1983). *The development of mathematical thinking*. New York: Academic Press.

Fuson, K. C. (1988). *Children's counting and concepts of number*. New York: Springer-Verlag.

Lu, C., Bloom, J., & Frye, D. (1988). *Mastering the machine: A comparison of the mouse and touchscreen for children's use of computers.* Unpublished manuscript.

Siegler, R. S., & Jenkins, E. (1989). *How children discover new strategies*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sleeman, D., Kelly, A. E., Martinak, R., Ward, R. D., and Moore, J. L. (1989). Studies of diagnosis and remediation with high school algebra students. *Cognitive Science* 13: 551-568.

Steffe, L. P., & Cobb, P. (1988). *Construction of arithmetical meanings and strategies*. New York: Springer-Verlag.

Stevenson, H. W., Lee, S. Y., & Stigler, J. W. (1986). Mathematics achievement of Chinese, Japanese, and American children. *Science* 231: 693-699.

Uttal, D. H., Lummis, M., & Stevenson, H. (1988). Low and high mathematics achievement in Japanese, Chinese, and American elementary-school children. *Developmental Psychology* 24: 335-342.

Factors Determining When to Interrupt and What to Say: An Empirical Study of the Case-Method Tutoring Approach

Deborah K. Galdes
Silicon Graphics Computer Systems
2011 North Shoreline Boulevard, Mountain View, CA 94039-7311

Philip J. Smith
Cognitive Engineering Research Lab
Department of Industrial and Systems Engineering
The Ohio State University, Columbus, OH 43210

Jack W. Smith, Jr.
Department of Pathology
The Ohio State University, Columbus, OH 43210

Abstract: Two important issues in the design of an intelligent coaching system are when to interrupt and what to say. This paper describes an empirical study of human tutors which was carried out to answer these specific questions. The tutors in this study used a "hands-on" tutoring approach to teach students in the medical problem-solving domain of immunohematology. The knowledge acquisition method represents a unique approach to collecting the necessary empirical data, and has several advantages over other similar techniques.

The results demonstrate that tutors take many factors into account when deciding when to interrupt and what to say, such as whether the student is likely to come across information in the near future that will allow him to recognize his error. This paper discusses several of these factors and how these factors influence the design of an ITS. The results are currently being used as the basis for designing an ITS for this same domain.

Introduction

If asked, "What are the important issues in designing an intelligent coaching system?", most designers of these systems would automatically respond, "When to interrupt and what to say." (Burton & Brown, 1979; Breuker, 1988). But although designers agree on the questions, they neither agree on the answers nor where to look for these answers.

For example, two well-known coaching systems which demonstrate very different philosophies for "when to interrupt" are Anderson's LISP tutor (Anderson, Boyle & Reiser, 1985) and the WEST tutor (Burton & Brown, 1979). Their different approaches partially result from using different bases for specifying the design of the tutor. That is, the LISP tutor is based on the ACT* theory of how students learn (Anderson, Boyle & Reiser, 1985), and this theory implies that students should be given immediate feedback on all errors. In contrast, WEST is based on Burton and Brown's personal philosophy of coaching which states that students should not be interrupted for every error (Burton & Brown, 1979).

An alternative basis for design, and one rarely used, is to rely on a descriptive theory of human tutoring. This approach has been used successfully to specify the design of ITSs which use the Socratic method to teach factual knowledge (Collins, Warnock & Passafiume, 1975) and inferential skills (Stevens & Collins, 1980). It has not, however, been a prevalent method for specifying the design of ITSs which teach procedural knowledge using the case method (Clancey, 1979) (i.e., "hands-on" problem solving). This is demonstrated by the lack of literature on this topic.

This paper, then, discusses an empirical study of human tutors who used the case method to teach students in the medical problem-solving domain of immunohematology (blood banking). Specifically, it discusses several factors which human tutors seem to use in deciding when to interrupt and what to say, and how these factors influence the design of an ITS. These factors are currently being used to design an ITS for this same domain of blood banking.

Methods

We viewed the problem of discovering a tutor's expertise as a knowledge acquisition task (Kidd, 1988). Both our data collection and analyses methods are summarized below and described in detail in Galdes (1990).

The Knowledge Acquisition Method

Several pilot studies led us to the final design described below and depicted in Figure 1.

1) The tutor and student were in separate rooms with limited visual communication over a video link and controlled audio communication over an audio link. This setup differs from previous studies of human tutors which all used computer terminals as the communication link when the tutor and student were in separate rooms.

2) The tutor and student shared visuals over the video link. The cameras used to support this link were positioned to give the tutor and student a "bird's-eye view" of the other's hands and worksheets, and nothing else (i.e., not the other person's face). The tutor always had visual access to what the student was currently working on. The student only received visual information from the tutor when the tutor had specific drawings or diagrams to show the student. (This rarely occurred.)

3) The default situation was that the tutor was watching the student solve the problem via the video link and the audio link was disconnected. When either the student or the tutor wanted to initiate a conversation, they would press a button and the audio link would be connected. At this time, the tutor and student could talk freely to each other. When the conversation was finished, the audio link was disconnected. Whenever this link was disconnected, the tutors and students were instructed to "think outloud."

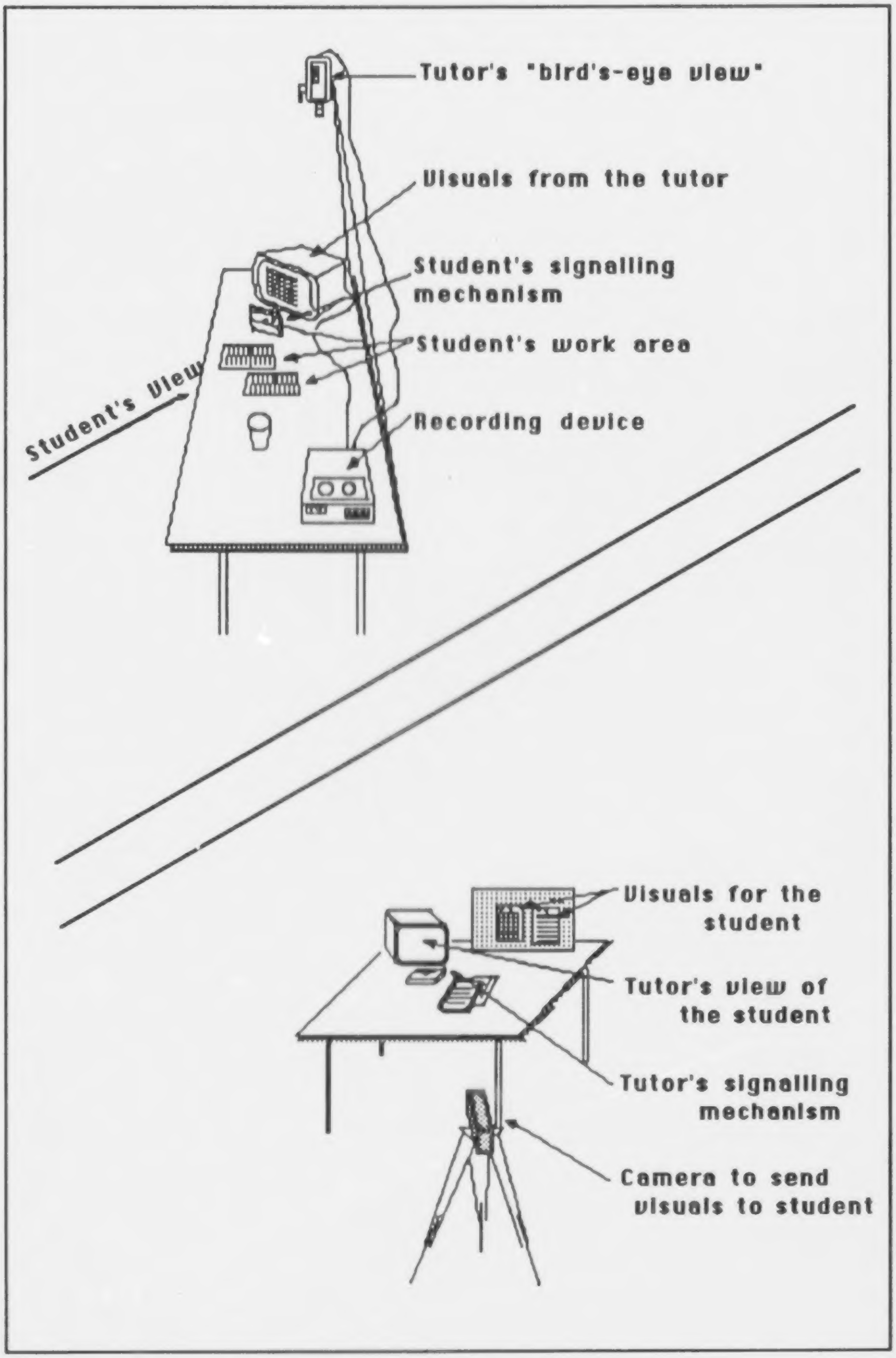

Tutor's "bird's-eye view"
Visuals from the tutor
Student's signalling mechanism
Student's work area
Student's View
Recording device
Visuals for the student
Tutor's view of the student
Tutor's signalling mechanism
Camera to send visuals to student

Figure 1. Students' and tutors' view of the world.

Our methodology demonstrated several important advantages. First, this methodology reduced the amount of nonverbal communication, yet retained face validity of the data by allowing tutors and students to carry on natural tutoring dialogues rather than typing at terminals. Second, it allowed us to, at times, collect verbal protocols from the tutors and students, giving us more insight into their cognitive processing. Third, tutors and students solved the cases using a familiar paper-and-pencil approach. This meant that we did not have to build a human-computer interface before we knew what capabilities tutors and students required of such a system.

Selecting Cases

We used five cases in our study. The cases were structured so that the students would have been taught all of the concepts necessary to solve them. The tutor's job was to help students pull all of this knowledge together when solving actual cases.

Subjects

Our subjects included two tutors and ten students. Both tutors were practicing experts in the domain of immunohematology and had experience tutoring various levels of students and trainees. The students were enrolled in an introductory blood banking course. Both the tutors and students received training on aspects of the task with which they were not initially familiar (e.g., thinking outloud).

Tutoring Sessions

Tutors and students were not given any specific instructions on how to behave other than that the student would solve cases and the tutor's goal was to make sure that students could solve problems like these on their own when they had completed the tutoring session. Both the tutor and student could initiate conversations during the tutoring sessions. During these sessions, each student solved the five cases. These sessions lasted from 2-3 hours per student.

In the case method, the student plays the role of the practitioner and begins each problem with the patient's case history. He is then expected to ask for further information and lab test results that he thinks will provide the necessary confirming and disconfirming evidence for coming to a conclusion. We provided a "lab" for our students to get this necessary additional information.

Data Analysis Procedures

After collecting the necessary empirical data, our task was to analyze it for "tutoring expertise." The first step of the analysis consisted of creating hard-copy transcripts from the audio and video tapes. These transcripts included the following data:

1) Verbal protocols which were collected while the tutor was deciding whether or not to interrupt the student.
2) Tutor-student dialogues which occurred after the tutor interrupted the student or the student requested help.
3) The students' actions as seen in the tutor's view of the world.

The second step was to select one of the two tutors to model in detail. Our intent when collecting data on two tutors was to model one tutor in detail and use the other tutor for a later comparison. We felt justified in this approach because we can expect different experts to use different strategies to achieve the same result (Simon, 1980; Kail & Bisanz, 1982).

Because we were interested in the questions of "when to interrupt" and "what to say," our third step was to mark all of the places in the transcripts where the tutor seemed to consider discussing a topic with the student, and to describe all of the events associated with each of these potential topics for discussion. We termed each potential topic and its associated events an "incident." The associated events consisted of:

- the student's action(s) at the time the topic arose,
- what else the student was doing at the time,
- what the tutor was verbalizing, and
- how this topic was eventually resolved (i.e., when it was dropped, discussed, or reactivated as a slightly different topic).

In all, we found 207 incidents for five students on four cases.

The fourth step of the analysis was to:

1) identify factors which seemed to influence the tutor's decisions of how to handle each topic which arose, and
2) develop rules which relied on these factors and parsimoniously described the tutor's behavior in the separate incidents.

This involved sorting and resorting the "incidents" and continually modifying the factors and rules until the categories felt comfortable and obvious (in hindsight!). (The rule sets are discussed in Galdes (1990).)

Results

From our analysis we could see that the tutor took many factors into consideration when deciding when to interrupt and what to say. Six of these factors are discussed below along with their implications for designing an ITS. A discussion of the remaining ten factors can be found in Galdes (1990).

The Tutor's Goal

We found two values for this factor:

- correct an action that represented a definite error in the students thinking, and
- identify if what appeared to potentially be an error in the student's thinking was truly a definite error.

The value of this factor appeared to be *the primary influence on what other factors the tutor seemed to consider.*

If the goal was to *correct* a definite error, the tutor seemed to primarily consider *different* *aspects about the specific error to correct*. These were factors such as the cause of the error (e.g., "forgetting" vs. garbled or missing knowledge) and the category of the error (e.g., skipping a goal). This conclusion is similar to that found in other studies of human tutors - tutors reason about categories of errors rather than individual errors when determining how to correct an error (Stevens & Collins, 1977; Littman, Pinto & Soloway, 1986).

In contrast, though, if the goal was to identify if a potential error was truly a definite error, the tutor seemed to mainly consider factors which told her *things about the student's current state* and *what information she could expect to get from future student actions*. These were factors such as the student's current goal, the student's current hypotheses, and whether or not additional information was available and likely from future student actions for resolving the question of whether this potential error was truly a definite error.

Implications. This factor implies that the student model and domain expert components must not only pass errors to the tutoring expert, but they must also pass the certainty of the particular error. This certainty, then, determines the tutoring expert's goal with respect to that error (i.e., correct or identify), which in turn determines what other factors to consider.

Whether or Not the Error was Caused by Forgetting and Information the Student is Likely to Come Across in the Near Future

These two factors taken together allowed the tutor to determine *whether or not she could use a "wait-and-see" strategy to allow the student to find and correct the error for himself*. That is, the tutor only appeared to use this "wait-and-see" strategy to achieve the goal of correcting errors if both of the following were true:

1) The tutor knew, based on past experience with this student, that the student had the correct knowledge, but for some reason failed to recall it at the appropriate time.
2) Future student actions would cause something to happen that would enable the student to notice his slip.

The tutor did not use this "wait" strategy when it appeared likely that nothing was going to happen that would allow the student to realize that he had made an error.

Implications. These two factors taken together give the tutoring expert component an additional strategy for correcting errors - wait-and-see if the student will recognize and correct the error himself. The second of these factors, however, presents several non-trival implications for the design of the student model and domain expert components.

First, it implies that the student model must be executable so that it can be run to determine what information this specific student will likely come across in the future. Second, it implies that the domain expert and student model must be able to reason about whether this information will be sufficient for the student to realize that he has made an error. Third, the domain expert and student model must be able to determine when the student has gone beyond the point of realizing his own error. That is, they must be able to tell the tutoring expert when the "wait-and-see" strategy has failed.

The Source of Inference for a Potential Error and the Current Context with respect to the Student's Solution Path

One source which the tutor seemed to rely on for identifying potential errors was her own generic student model of what students at this specific level of expertise typically did and did not know. Errors in this category represent potential topics which the tutor considered discussing, but which did not arise from any action or inaction on the student's part. In fact, many of these potential errors were unlikely to ever be detected from student actions. Examples of errors in this category include

- being able to use the causal models of the domain to predict the outcome of a specific lab result before receiving that result from the lab,
- knowing the goal behind an action,
- knowing the reason behind a specific goal, and
- knowing why the absence of a goal or action was correct in some cases.

Because this class of errors did not arise opportunistically from some incorrect or potentially incorrect action on the student's part, the tutor had to actively be on the lookout for appropriate contexts for discussing them. Thus, an additional factor which the tutor considered was the current context with respect to the students solution path.

Implications. Potential errors from the generic student model represent an important class of errors for ITSs for two reasons. First, many of the errors in this category for our tutor represented errors which were unlikely to be detected from student actions. Thus, they must be planned in advance because they will almost never arise opportunistically from a student's actions. Second, this class of errors implies that an ITS for this or similar domains cannot limit itself to only addressing definite errors. It must also sometimes bring up topics which represent potential errors if it is to fully tutor a student in the domain.

The second factor above, the current context of the solution path, implies that the student model and domain expert must have the knowledge of what constitutes an appropriate context for discussing each of these "generic" potential errors. Also, because such a context could arise more than once during a case or over the course of several cases, the student model and domain expert

- not only have to determine when an appropriate context has arisen for a specific potential error,
- but also *have to be able to predict if future contexts will arise for discussing this error.*

This allows the tutor to be much more flexible in deciding whether or not to discuss a potential error from the generic student model immediately or later.

The Likelihood that the Student Will Call Soon

The tutor seemed to consider this factor when the category of the definite error was that the student appeared generally confused - that is, the student was tapping his pen, continuously shuffling his lab results, or doing nothing for several seconds. In these instances, if it appeared that the student was likely to call soon, the tutor waited to interrupt.

This rule goes against a finding by Pomerantz (Fox, 1988) who analyzed naturally-occurring dialogues and found that "...if we see someone having difficulty in some way, it is preferred for us to offer help before that help is requested." One explanation for this inconsistency is that in tutorial dialogues, the tutor's primary goal is to help the student. She could have always jumped in with help as soon as the student showed any signs of having trouble. What seems to have been the preferred approach for our tutor, though, was to look for ways to limit the number of interrupts to the student. Letting the student call her was one way to reduce this number.

Implications. Again, this factor implies that an ITS needs an executable student model. In addition, it requires that this executable model must not only predict the students actions along the solution path, but must also predict his behavior for calling the tutor to ask for help.

Conclusions and Future Directions

In this paper, we have detailed several factors which tutors might consider when deciding when to interrupt and what to say. Many of these factors are not obvious from intuition or armchair speculation about what makes up tutoring expertise. In addition, we have discussed how these factors influence the design of an ITS.

We are currently implementing these results in an ITS which uses the case method to tutor blood banking students. Our hope is that by basing the design of our ITS on empirical studies of human tutors rather than our own personal theories of tutoring, it will take us less time to reach the stage of a useful and usable ITS.

In addition to building an ITS, we are also continuing our analyses of the tutoring transcripts from two perspectives. First, we are continuing our analyses to gain insight into additional aspects of tutoring expertise such as

- how to talk about a topic,
- what kinds of questions students ask, and
- how tutors respond to these questions.

Second, although the primary goal of this research was to identify tutoring expertise, these transcripts also appear to give us valuable insights into the student model and domain expert components. Thus, we are also beginning to analyze the transcripts from these other perspectives. As an example, we know that medical diagnosis differs from arithmetic in that it requires knowledge about the procedure of diagnosis plus an understanding of a causal model of the system being diagnosed (Clancey, 1986). Our data demonstrates that tutors appear to spend a good deal of the time discussing this causal model. If this turns out to be true, then we can use our empirical data to determine exactly when and why tutors use this model, and possible interface designs that will allow an ITS and student to converse about it.

Acknowledgement: This work has been supported by the National Heart, Lung, and Blood Institute under grant HL-38776.

References

Anderson, J.R., Boyle, C. & Reiser, B.J. (1985). Intelligent tutoring systems. *Science* 228: 456-462.

Breuker, J. (1988). Coaching in help systems. From *Artificial Intelligence and Human Learning, Intelligent Computer-Aided Instruction*, J. Self (Ed.). Chapman and Hall Computing.

Burton, R.R. & Brown, J.S. (1979). An investigation of computer coaching for informal learning activities. *International Journal of Man-Machine Studies* 11: 5-24.

Clancey, W.J. (1979). Tutoring rules for guiding a case method dialogue. *International Journal of Man-Machine Studies* 11: 25-49.

Clancey, W.J. (1986). Qualitative student models. *Annual Review of Computer Science* 1: 381-450.

Collins, A, Warnock, E.H., & Passafiume, J.J. (1975). Analysis and synthesis of tutorial dialogues. From *The Psychology of Learning and Motivation, vol. 9*, G.H. Bower (Ed.). Academic Press.

Fox, B.A. (1988). Cognitive and interactional aspects of correction in tutoring. *Technical Report #88-2*. Department of Linguistics and Institute of Cognitive Science, University of Colorado, Boulder.

Galdes, D.K. (1990). An empirical study of human tutors: The implications for intelligent tutoring systems. *Doctoral Dissertation*. The Ohio State University.

Kail, R.V. Jr. & Bisanz, J. (1982). Cognitive strategies. From *Handbook of Research Methods in Human Memory and Cognition*, C.R. Puff (Ed.). Academic Press.

Kidd, A. (Ed.) (1988). *Knowledge Acquisition for Expert Systems: A Practical Handbook*. Plenum Press.

Littman, D.C., Pinto,J. & Soloway,E. (1986). An analysis of tutorial reasoning about programming bugs. *Proceedings of the National Conference on Artificial Intelligence*: 320-326.

Simon, H.A. (1980). Cognitive science: The newest science of the artificial. *Cognitive Science* 4: 33-46.

Stevens, A.L. & Collins, A. (1977). The goal structure of a Socratic tutor. *Proceedings of the Association for Computing Machinery National Conference*: 256-263.

Stevens, A.L. & Collins, A. (1980). Multiple conceptual models of a complex system. From *Aptitude, Learning, and Instruction, Vol. 2: Cognitive Process Analyses of Learning and Problem Solving*, R.E. Snow, P.A. Federico & W.E. Montague (Eds.). Lawrence Erlbaum Associates.

Uses of Interactivity in Tutorial Support for Exploratory Learning

Peter Goodyear and Robin Johnson
Department of Educational Research
Lancaster University, Lancaster, LA1 4YL, England

Annet van Kouwen
Faculteit der Psychologie
Universiteit van Amsterdam, Weesperplein 8, 1018 XA Amsterdam, The Netherlands

Tamara Ronteltap
Faculteit der Pedagogische, Andragogische en Onderwijskundige Wetenschappen
Universiteit van Amsterdam, Prinsengracht 227, 1015 DT Amsterdam, The Netherlands

Abstract: This paper reports some results of experimental work carried out on the European Community-Funded DELTA project SIMULATE. The goal of the work was to model some of the tutoring methods used by human teachers while supporting the use of simulation programs.

Use of the simulation programs can be characterised as "exploratory learning," in the sense that the learner is finding paths through a rich and relatively complex information resource while exercising a high level of control over the direction of their search. Exploratory learning demands and develops certain valued higher-order cognitive skills. In contrast with some other, perhaps better understood, learning situations, the learner spends a relatively high proportion of their time and cognitive resources in goal setting. Their goals are typically fragile and fuzzy. There is abundant empirical evidence to demonstrate that unsupported exploratory learning is often wasteful and frustrating for the learner. However, unsophisticated tutorial support can easily distort, or even destroy, the valued central features of exploratory learning. Maintaining a tutorial interaction which can support, but not subvert, exploratory learning is a non-trivial task and the methods used by tutors are not intuitively obvious nor readily derivable from instructional theory.

The paper reports findings from concurrent and retrospective think-aloud studies of human tutoring. The studies were made using single learner-subjects and either single or paired tutor-subjects. In each case, the learner was working with a quantitative dynamic systems simulation. Tutorial support was given either face-to-face or over a computer-mediated link (to permit concurrent verbalisation). The report focuses on special features of interactive tutoring. We contend that few intelligent tutoring systems projects have paid much attention to interactivity in tutoring. Even less attention has been paid to the advantages that can be obtained from understanding some of the uses to which interactivity may be put. We identify a number of such uses. We offer an account of tutors' knowledge about the repairability of interactions and about the provisional status of utterances and actions in negotiating shared understandings of the learning and tutoring tasks at hand.

Designing Educational Software Around Questioning

Arthur C. Graesser and Mark C. Langston
Department of Psychology
Memphis State University, Memphis, TN 38152

Kathy L. Lang
MRE 2M-329
Bellcore
445 South St., Morristown, NJ 07960

Abstract: Researchers in AI, cognitive science, and education have emphasized the importance of questioning in the process of text comprehension, learning, and creativity. However, the existing educational software is not at the stage where students can quickly ask a broad diversity of questions and quickly receive informative answers. We have designed a "Point and Query" (P&Q) interface which is designed to overcome such barriers. The student (1) points to a word, phrase, or picture element on the computer screen that displays the material to be learned, (2) points to one of a set of questions that is relevant to the element, and (3) reads the answer to the question. The P&Q interface is integrated with a hypertext environment so answers to questions are preformulated and quickly accessed. We have developed a learning module that handles taxonomic, spatial, and causal knowledge structures in the context of woodwind instruments. The available questions and the strategies of answering questions are based on psychological models that are computationally tractable.

Our goal is to design educational software that facilitates an inquisitive, creative thinking style in the mind of the student. More specifically, the student should be able to ask questions very quickly and should receive informative answers from an expert very quickly. Researchers in AI, cognitive science, and education have frequently emphasized the importance of question asking and question answering in the process of text comprehension, learning, and creativity (Collins, 1988; Palincsar & Brown, 1984; Schank, 1986). However, existing question answering systems are not at the stage where students can ask a broad diversity of questions very quickly (i.e., within two seconds) and receive informative answers very quickly (i.e., within two seconds). We have developed a "Point & Query" (P&Q) interface that is designed to overcome these barriers.

There have been frequent attempts in computer science and artificial intelligence to make the process of question asking easier (Lang, Graesser, Dumais, & Kilman, in press). The earliest formats for asking questions involved a rigid query language; the words and syntax of the queries needed to be entered in a precise way. A "Structured Query Language" (SQL) was developed to make the commands and queries more similar to English, but the expressions were still cryptic, difficult to learn, and awkward for non-programmers. In the best of all possible worlds, the user would be able to ask questions in natural language.

However, it is widely acknowledged that it would be extremely difficult, if not impossible to develop a computer system that understands natural language (Winograd & Flores, 1986). Therefore, systems have been designed with special facilities that fall somewhere between the extremes of a rigid query language and natural language: Query by Example (Zloof, 1975), RABBIT's "retrieval by reformulation" (Williams, 1984), Texas Instrument's menu-driven natural language interface (Tennant, 1987), and Latent Semantic Indexing (Deerwester, Dumais, Landauer, Furnas, & Harshman, 1990).

Unfortunately, the existing interfaces for asking questions have several shortcomings which prevent them from being serious contenders for educational software. Each suffers from one or more of the following problems.

1. It takes a long time for the computer user to ask a question (often several minutes).
2. The computer doesn't handle questions that the user wants to ask.
3. The system misinterprets questions.
4. The system does not provide quick feedback on the repertoire of questions it can handle.
5. The knowledge base is not organized in an intelligent manner which supports dialogue and information exchange.
6. The system is difficult to learn.

Just as there have been attempts to improve *question asking* facilities on computers, there have been attempts to improve *question answering* facilities. Sophisticated "help" systems are available on most computers; help systems may be construed as a type of question answering facility. Users of expert systems and intelligent tutoring systems can ask and receive answers to why, how, what-if and other types of questions (Buchanan & Shortliffe, 1984; Sleeman & Brown, 1982; Wenger, 1987). There are several models of question answering in artificial intelligence (Allen, 1983; Lehnert, 1978; Schank & Abelson, 1977). Unfortunately, however, many of these question answering facilities are not based on theories of human question answering. As a consequence, users become quickly frustrated and eventually abandon them. For example, the answers to why-questions in rule-based expert systems provide a trace of successful rules that lead to a particular conclusion. A trace of these rules is very unnatural for humans to use and appreciate. In general, the existing question answering facilities suffer from three problems:

1. The answers are not informative and relevant from the perspective of the human user.
2. It takes a long time for the computer to answer the question.
3. There is too much information in the answer.

The "Point and Query" (P&Q) Interface

We have developed a P&Q interface which corrects most, if not all of the problems with the above systems that have integrated questioning in computer software. According to the P&Q interface, the student can both ask a question and receive an answer within 2 seconds by executing the following three actions.

- ***Point to a word, phrase, or picture element on the computer screen.*** It would be ideal to have a touch panel screen, but our current interface has a mouse.
- ***Point to one question*** among a set of questions that is displayed in a question box. The set of questions depends on the element specified in step 1.

- ***Read the answer to the question***, which the computer presents within one second. The P&Q interface is integrated with a hypertext environment so answers to questions are preformulated and accessed quickly.

The set of questions and the answers to each question are based on models of human cognition. Therefore, the P&Q interface provides the capability of a question-answer dialogue between a student and an expert teacher.

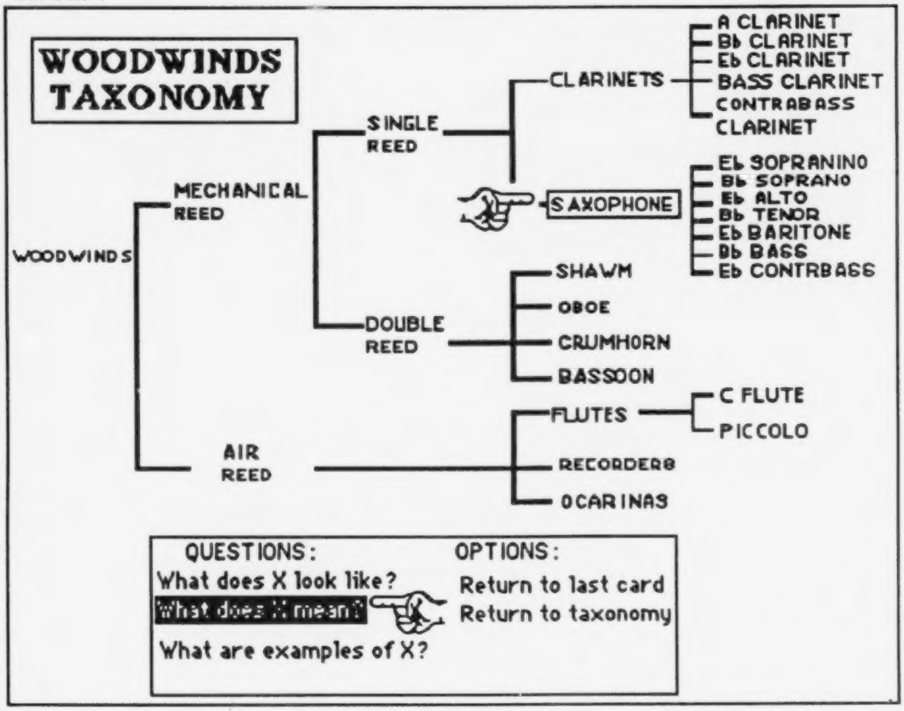
Figure 1 presents an example of a single question that is asked in the context of woodwind instruments. Figure 1 shows two computer screens which would exist at two successive points in time. The top screen depicts a point during the interactive session in which the taxonomy of woodwind instruments is presented. The words in capital letters can be pointed to and queried whereas lower case words cannot. With this simple rule, the user knows exactly what screen elements can be unpacked with questions. The top screen also illustrates how a question is asked. The user first points to *SAXOPHONE*, the screen element that the user wants to learn more about. Once this element is declared, the computer presents a set of questions in the "question box." The set of questions depends on conceptual properties of the element, as will be discussed later. The user points to the question *What does X mean?*, which denotes the question "What does saxophone mean?" It takes very little effort to ask the question: the user simply points to the screen in two successive strokes. Moreover, the questioner immediately knows what questions the computer system can handle.

The bottom screen contains the answer to the question *What does saxophone mean?* The answer is formulated on the basis of a model of human question answering called QUEST (Graesser & Franklin, 1990), as will be discussed later. The user could then ask questions about any element in the answer which is printed in capital letters. For example, the user could point to REED and then point to one of the questions that can be asked about REED. This process continues indefinitely until the user is satisfied. There also is an "option box" which permits the user to exit from the system and to perform special-purpose functions; the user simply points to one of the options in the option box instead of pointing to a question in the question box.

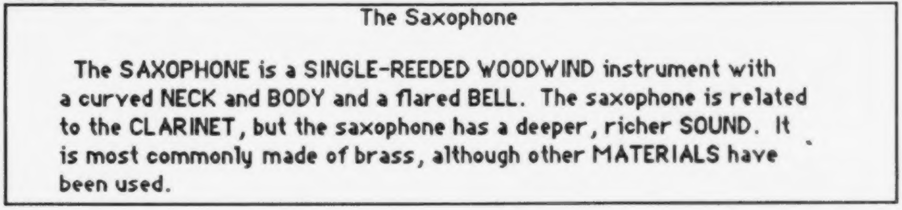
The P&Q interface corrects nearly all of the problems with previous questioning facilities. Consider the six problems that previous facilities had with question asking. Unlike the previous facilities, in the P&Q interface it takes very little time and effort to ask a question (i.e., just two points of a finger). The computer handles questions that the user wants to ask because the questions associated with each element are based on a model of human question generation (Graesser, Person, & Huber, in press) and (if time, energy, and money permits) on empirical research involving human learners. The system doesn't misinterpret questions because there is a closed set of questions associated with each element, all of which the user can observe in the question box. The system immediately conveys to the user exactly what set of questions it can handle, namely those questions in the question box. The knowledge base is organized in an intelligent manner that supports question answering. The interface is very simple, so it is easy to learn how to use.

Regarding question answering, the P&Q interface corrects the three problems that we identified as being associated with previous question answering facilities. First, the P&Q interface delivers relevant, informative answers because the answers are based on a model of human question answering called QUEST (Graesser & Franklin, 1990). Second, the answers are formulated very quickly because the system is implemented in a hypertext environment. Third, there is rarely too much information on the screen because an answer addresses a particular question, on a single computer screen.

Screen 1


WOODWINDS TAXONOMY
WOODWINDS
MECHANICAL REED
SINGLE REED
CLARINETS
A CLARINET
B♭ CLARINET
E♭ CLARINET
BASS CLARINET
CONTRABASS CLARINET
SAXOPHONE
E♭ SOPRANINO
B♭ SOPRANO
E♭ ALTO
B♭ TENOR
E♭ BARITONE
B♭ BASS
E♭ CONTRBASS
DOUBLE REED
SHAWM
OBOE
CRUMHORN
BASSOON
AIR REED
FLUTES
C FLUTE
PICCOLO
RECORDERS
OCARINAS
QUESTIONS:
What does X look like?
What does X mean?
What are examples of X?
OPTIONS:
Return to last card
Return to taxonomy

Screen 2



The Saxophone

The SAXOPHONE is a SINGLE-REEDED WOODWIND instrument with a curved NECK and BODY and a flared BELL. The saxophone is related to the CLARINET, but the saxophone has a deeper, richer SOUND. It is most commonly made of brass, although other MATERIALS have been used.

Figure 1.

A Model of Human Question Generation

As mentioned above, the questions associated with each element are based on a model of human question generation (Graesser, Person, & Huber, in press). Some of these mechanisms exist in computer models of question generation, problem solving, and reasoning, such as AQUA (Ram, 1990; Reisbeck, 1988), SWALE (Schank, 1986), and SOAR (Laird, Rosenbloom, & Newell, 1987). Other mechanisms exist in theories of conversation (Allen, 1983; Clark & Schaefer, 1989), education (Collins, 1988), and question generation per se (van der Meij, 1987). In addition to these theoretical contributions, Graesser, Person, and Huber (in press) discovered question generation mechanisms when they analyzed transcripts of tutoring sessions on a variety of topics (e.g., research methods, mathematics, computer systems, climate, agriculture).

One critical challenge is to determine which set of questions is presented in the question box when the user points to a particular element (e.g., SAXOPHONE). On the current P&Q system, the question set is based on two variables. First, there is the category of the element: a noun or noun-phrase (e.g., SAXOPHONE), a verb or adjective referring to an involuntary event (e.g., the note becomes SHARP), a verb or verb-phrase referring to an intentional action (e.g., the player TIGHTENS the embouchure), versus an adjective referring to a state (e.g., the reed is FLEXIBLE). Second, there is the category of knowledge structure that underlies the elements: taxonomic hierarchy, causal network, goal/plan hierarchy, versus spatial network (Graesser & Clark, 1985).

There is a unique set of question categories associated with each cell in the matrix (element category X knowledge structure category). For example, the following questions are associated with the nouns in a taxonomic hierarchy: What does X mean?, What are the properties of X? , What does X look like?, and What are types of X?. In contrast, the following questions are asked about events in causal structures: Why did X occur?, How did X occur?, What if X didn't occur?, and What are the consequences of X occuring?.

Another challenge lies in the selection of screen elements that can be queried. Such decisions are based on three considerations. First, we consider what type of knowledge structure is critical. If the prevailing knowledge structure involves a taxonomic hierarchy, then nouns would tend to be selected; if the prevailing knowledge structure involves a goal/plan hierarchy, then intentional action verbs would tend to be selected. Second, we consider what knowledge is available in the learning module. Obviously, we do not unpack an element if there is very little to say about the element. Third, we consider the amount of knowledge that learners would have about the topic. That is, the learner's knowledge base may be incomplete or erroneous, so these knowledge deficits would need to be corrected. The typical learner's knowledge deficit may involve an obstacle in planning, a glitch in an explanation, a gap in critical knowledge, or a contradiction (see Graesser, Person, & Huber, in press).

A Model of Human Question Answering

The answers to the questions are designed around a model of human question answering called QUEST (Graesser & Franklin, 1990; Graesser, Hemphill, & Brainerd, 1989; Graesser, Lang, & Roberts, in press). QUEST handles a broad diversity of questions in the context of taxonomic, causal, goal-oriented, and spatial knowledge structures. QUEST can account for the answers that adults produce when they answer questions and for "goodness-of-answer" ratings that adults give to question-answer pairs.

The QUEST model represents knowledge in the form of a "conceptual graph structure", with nodes (i.e., nouns, events, actions, states) that are connected by directed, relational arcs (e.g., Reason, Manner, Outcome, Property, etc.). Each question category has a particular "arc search procedure" that pursues specific paths of arcs and nodes in an information source. For example, when an intentional action is probed with a why-question, then legal answers are on paths of forward Reason arcs (i.e., superordinate goals) but not backward Reason arcs (i.e., subordinate actions) and not arcs in most other categories. The arc search procedure associated with a particular question category ends up converging on a small set of nodes from a particular information source. The full-blown QUEST model is more complex than the mechanism of arc search procedures operating on conceptual graph structures; however, such a mechanism has been implemented in the P&Q interfaces that we have developed so far.

Final Comments

In closing, we believe that the P&Q interface will encourage a more active, inquisitive, and creative thinking style in the minds of learners. We imagine a learner asking hundreds of questions in an hour session with a learning module. Students will learn how to ask good questions about topics they are very unfamiliar with. This could have a radical impact on the cognitive style of individuals in our culture and in other cultures.

This research was funded by grants awarded to the first author by the Office of Naval Research (N00014-88-K-0110 and N00014-90-J-1492).

References

Allen, J. (1983). Recognizing intentions from natural language utterances. In M. Brady and R.C. Berwick (Eds.). *Computational models of discourse.* Cambridge, MA: MIT Press.

Buchanan, B.G. and Shortliffe, E.H. (Eds.) (1984). *Rule-based expert systems.* Menlo Park, CA: Addison-Wesley.

Clark, H.H. and Schaefer, E.F. (1989). Contributing to discourse. *Cognitive Science* 13:259-294.

Collins, A. (1988). Different goals of inquiry teaching. *Questioning Exchange* 2: 39-46.

Deerwester, S., Dumais, S.T., Landauer, T.K., Furnas, G.W. and Harshman, R.A. (1990). Indexing by latent semantic analysis. *Journal of the Society for Information Science* 141(6):391-407.

Graesser, A.C. and Clark, L.F. (1985). *Structures and procedures of implicit knowledge.* Norwood, NJ:Ablex.

Graesser, A.C. and Franklin, S.P. (1990). QUEST: A cognitive model of question answering. *Discourse Processes* 13:279-303.

Graesser, A.C., Hemphill, D.D., and Brainerd, L.E. (1989). Question answering in the context of causal mechanisms. In *Proceedings of the 11th Annual Conference of the Cognitive Science Society*, 621-626. Hillsdale, NJ: Erlbaum.

Graesser, A.C., Lang, K.L., and Roberts, R.M. (in press). Question answering in the context of stories. *Journal of Experimental Psychology* : General.

Graesser, A.C., Person, N.K., and Huber, J.D. (in press). Mechanisms that generate questions. In T. Lauer, E. Peacock, and A.C. Graesser (Eds.), *Questions and information systems.* Hillsdale, NJ: Erlbaum.

Laird, J.E., Rosenbloom, P.S., and Newell, A. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence* 33(1):1-64.

Lang, K.L., Graesser, A.C., Dumais, S.T., and Kilman, D. (in press). Question asking in human-computer interfaces. In T. Lauer, E. Peacock, and A.C. Graesser (Eds.), *Questions and information systems.* Hillsdale, NJ: Erlbaum.

Lehnert, W.G. (1978). *The process of question answering.* Hillsdale, NJ: Erlbaum.

Palinscar, A.S. and Brown, A.L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction* 1(2): 117-175.

Ram, A. (1990). Decision models: A theory of volitional explanation. *Proceedings of the 12th annual conference of the Cognitive Science Society*, 198-205.

Reisbeck, C.K. (1988). Are questions just function calls? *Questioning Exchange* 2(1):17-24.

Schank, R.C. (1986). *Explaining problems: Understanding mechanically and creatively.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. and Abelson, R.P. (1977). *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures.* Hillsdale, NJ: Erlbaum.

Sleeman, D.H. and Brown, J.S. (Eds.) (1982). *Intelligent tutoring systems*. New York: Academic Press.

Tennant, H.R. (1987). Menu-based natural language. In S.C. Shapiro and D. Eckroth (Eds.), *Encyclopedia of artificial intelligence*, 594-597. New York: John Wiley & Sons.

Wenger, E. (1987). *Artificial intelligence and tutoring systems.* Los Altos, CA: Morgan Kaufman.

Williams, M.D. (1984). What makes RABBIT run? *International Journal of Man-Machine Studies* 21:333-352.

Winograd, T. and Flores, C.F. (1986). *Understanding computers and cognition*. Norwoord, NJ: Ablex.

van der Meij, H. (1987). Assumptions of information-seeking questions. *Questioning Exchange* 1:111-117.

Zloof, M.M. (1975). Query by example. *Proceedings of the National Computer Conference* 44:431-438. Arlington, VA:AFIPS Press.

Environments for Situated Conceptual Learning

James G. Greeno
Stanford University and the Institute for Research on Learning
Palo Alto, CA 94304, USA

Abstract: I discuss some theoretical ideas and distinctions about conceptual reasoning that are consistent with the framing assumption that cognition and learning are situated in material and social settings. Concepts are characterized as sets of conceptual entities and constraints that are used in making inferences in situations and in constructing mental models that simulate events in order to support inferences about situations that are not present. On this view, conceptual reasoning is a part of social practice, and learning to reason conceptually occurs through participation in the activities of conceptual reasoning, especially in learning to construct and reason with mental models that are shared in a community of practice.

This paper includes a theoretical discussion of conceptual reasoning and learning that adheres to the view that cognition is situated activity. The theoretical ideas are relevant to the way we think about designs of learning environments, including the functions of computational systems as resources for learning.

I begin with a sketch of some examples of conceptual reasoning that illustrate desirable abilities that should be strengthened by students' learning in schools. Second, I briefly describe a theory of conceptual reasoning as activity in situations, in which discourse about conceptual entities in the situation supports coordination of activities and shared understanding of actions and events. Third, I discuss reasoning with mental models, an aspect of social practice that enables groups and individuals to reason about nonpresent situations, including situations in which the objects are abstract entities that correspond to structural features of ordinary situations. Fourth, I present a characterization of learning to reason conceptually that follows from the view of reasoning and a conjecture about conditions that I expect should be favorable for that learning, based on the theoretical ideas. Finally, I present some suggestions for the design of learning environments that are implied by the theoretical ideas and the conjecture about learning.

Examples of Conceptual Reasoning

What do we hope that students will be able to do with the concepts they learn in school? I mention a few abilities that seem valuable, among many that could be listed.

We would like students to be able to participate in conversations that are informed by concepts. We would like students to benefit from understanding of concepts in the work they do on their jobs or in other organizations, including analytic projects and plans for proposed group activities. We would like students who understand a concept to be able to explain it to other people, and we would like students' understanding of concepts to support their further learning, including understanding of information that the concepts help make meaningful and learning of other concepts that are related to those that were learned earlier.

Consider a few hypothetical examples. A dinner conversation is concerned with water use in California, including topics like (1) some quantitative projections of population and the effect of population on water use; (2) year-to-year variation in the availability of water, (3) the advantage of drip irrigation to sprinklers in agriculture, the economic cost of changing to a new technology, and alternative ways of providing incentives to farmers. Concepts needed for this conversation include mathematical functions, statistical random variables, physics of evaporation, and the economic theory of decision making.

Other examples of the kinds of activity that we would like students to learn to participate in include ability to perform an analysis for a volunteer organization or one's company of likely impacts of various options in legislative bills for water conservation, to supervise students in a high-school class in projects about water conservation, or succeeding in college courses in ecology, economic geography, or decision science.

These activities of conceptual reasoning are embedded in social practice. Some of the target activities occur in conversations with other people. Some, like projects done for one's company or another organization, involve a mixture of committee work and individual research and writing, but are mainly done for socially determined purposes. I will propose a conjecture that the learning of concepts mainly occurs in contexts of social activity, and that understanding of concepts therefore originates primarily in social activity. I also will propose a characterization of concepts as collections of constraints and conceptual entities, and of conceptual reasoning as reasoning that depends on people's attunement to the constraints and representation of the conceptual entities that constitute a concept.

Conceptual Reasoning in Situations

The view of concepts that I present is coherent with a general view of cognition as situated with material objects and systems and with other people. Cognitive processes are, in this view, interactions with materials and other people, rather than operations on symbols contained in an individual's mind. Joyce Moore and I are are working on a theory of situated conceptual reasoning. Here, I sketch a few of the main distinctions.

We use a situation-theoretic idea of *states of affairs* (Barwise, 1989; Devlin, 1988) to formulate our hypotheses. A state of affairs is a set of properties and relations that can hold in a situation. We distinguish between material states of affairs, which hold in the situation, and registered states of affairs, which correspond to representations that may be written down, stated in conversation, or merely represented cognitively. A registered state of affairs may be a representation of a material state of affairs that holds in the situation, registered in a process of perception, or it may be inferred based on information in the material situation or based on symbolic expressions of previously registered states of affairs.

We use the term *reasoning* to refer to inferences that generate new registered states of affairs. An inference can occur because a person is *attuned to a constraint,* thereby being cognizant of a potential state of affairs that he or she then registers, or it can occur via a *symbolic inference,* in which previously registered states of affairs are the conditions for a rule-like inference of a symbolic representation of another state of affairs. Constraints are relations between *types of situations,* which are designated by parameterized states of affairs.

Moore and I characterize a *concept* as a collection of constraints and *conceptual entities.* Conceptual entities are objectifications of properties and relations in situations, in Smith's (1987) sense. Conversation that refers to conceptual entities can direct a person's attention to properties and relations that are involved in constraints that the person is attuned to, or to constraints that the person can become attuned to.

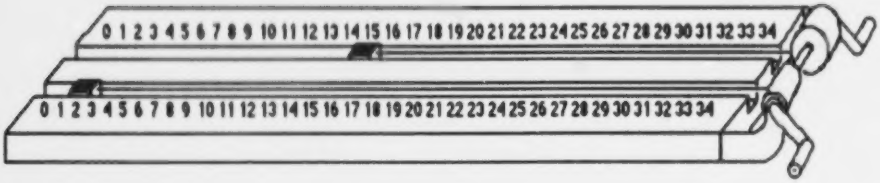


Figure 1. Winches used in experiments.

To illustrate these ideas, I will briefly describe some observations and an analysis that Moore and I have conducted. A more detailed description is given by Moore and Greeno (1991). We are studying performance in a situation that we designed, adapted from one used by Piaget, Grize, Szeminska, and Bang (1968/1977) to study children's understanding of quantitative functions. There are two tracks, each of which has a metal block that can be pulled by a string as the handle is turned. Each track behaves according to the formula $y = mx + b$, where b is the initial location of the block, m is the circumference of the spool, x is the number of turns of the handle, and y is the final location of the block. The sizes of the spools and the initial locations of the blocks are variable. We have interviewed students before they studied algebra, as well as some students who had either one or two years of algebra study. Our interviews included a variety of problems and questions, most of which involved comparisons between the two tracks, corresponding to solving a pair of linear equations. Here, we consider a simpler task, involving only one track.

Suppose that the block starts at 7, and the spool circumference is 3. The task is to answer the question, "How many turns will it take for the block to be at 25?" We observed four different methods of answering this question:

(a) by turning the handle until the block reaches 25, counting the number of turns it takes;
(b) by measuring successive intervals of 3 inches each along the ruler, counting the intervals until 25 is reached;
(c) by making a table of the locations of the block after different numbers of turns, until a number of turns corresponding to 25 is written;
(d) by subtracting the number of the initial location from the number of the final location, and dividing that distance by 3.

Different material states of affairs hold when these different methods are used. For example, when the block is moved by turning the handle, the following material states of affairs hold: (i) <block starts at 7>, (ii) <block ends at 25>, (iii) <handle turns 6 times>, (iv) <block is at 10 after 1 turn>, (v) <block is at 13 after 2 turns>,..., (vi) <block moves 3 inches each turn>, (vii) <location of block after each turn is 3 greater than before that turn>, (viii) <number of location is number of previous location plus 3>, (ix) <ending location is (6 times the spool size) plus the initial location>. When the question is answered by turning the handle, the first three states of affairs need to be registered. SOA (iii) is the answer to the problem, but it is obtained by counting the turns, rather than by an inference.

Method (b) involves an inference about the effects of turns, based on a physical simulation of turning. Material states of affairs that hold include: (i) <finger starts at 7>, (ii) <finger ends at 25>, (iii) <finger at 25 after 6 increments>, (iv) <finger at 10 after 1 increment>, (v) finger at 13 after 2 increments>,..., (vi) <finger moves 3 inches each increment>, (vii) <location of finger after each increment is 3 greater than before that turn>, (viii) <number of location is number of previous location plus 3>, (ix) <ending location is (6 times the spool size) plus the initial location>.

In these processes of generating states of affairs, conceptual entities include numbers. Constraints include generalizations of the states of affairs numbered (vi)-(ix). In Method (a), all these states of affairs hold because of the physical situation. In Method (b), (vi) holds because of the students. In Method (c), where a table is used, (viii) holds because of the students, and in Method (d), (ix) holds because of the students. In the various methods, then, different constraints that are part of the concept of a linear function are supplied by the students' being attuned to them.

Conceptual Reasoning with Mental Models

We follow Fauconnier (1985) and Johnson-Laird (1983) in assuming that when people communicate about a situation different from the one they are in, they construct mental models of the situation, with objects that correspond to the objects that are described and with properties and relations that either are designated in the communication or that the objects are known to have. We follow Yates (1985) in assuming that reasoning about nonpresent situations can occur through enactment of simulated events with the objects in the mental models that people can construct. We further assume that mental models work in simulating events because the symbolic objects that they include are exemplifications of conceptual entities, and that events that are simulated occur according to relevant constraints that apply because of the properties of the symbolic objects of the model.

I hypothesize that the conceptual entities and constraints that are shared in a community are an important means by which the members of the community construct shared information, meanings, understandings, opinions, and beliefs. The hypothetical examples that I mentioned in Section 1 of reasoning about water use in conversations, analysis projects, teaching, and learning, all involve thinking and communicating about situations that the people are not in. According to this hypothesis, the reasoning that occurs in these examples uses mental models in which the people enact simulations of systems as well as make inferences based on symbolic expressions that they express in conversations or in writing.

Learning Practices of Conceptual Reasoning

I hypothesize that concepts are learned primarily in activities of participating in discourse that includes construction and reasoning with mental models that include exemplifications of the conceptual entities and constraints that constitute the concepts. Conceptual learning occurs by processes of conversational negotiation, in which properties of objects in an individual's mental models are tested by simulating events and describing results that are either in agreement or disagreement with inferences that other individuals describe, and processes of conversational repair are used to improve alignment of terms with events in the simulations that different participants enact, and to effect changes in the behavior of mental models so that they produce results in better agreement.

I conjecture that favorable conditions for learning concepts may include group activities including interactions with physical or computational models that exemplify the conceptual entities that the community uses in its shared mental models. Interactions with physical or computational models can include experience of constraints that need to be incorporated into mental models for their successful use in simulating events in the domain. Discourse about the behavior of a model can direct attention to the objects in the model that exemplify conceptual entities, and to the behavior that exemplifies the constraint relations that need to be incorporated into the behavior of mental models. Simply asserting the constraints should be less effective, according to this view, because assertions of constraints would be less likely to evoke processes of constructing and reasoning with mental models.

A successful outcome of conceptual learning is internalization of the referents of discourse in the form of conceptual entities and constraints on their behavior that the person can exemplify in mental models that can be used to simulate events in order to make inferences about properties of events and their outcomes.

Implications for Design of Learning Environments

The discussion that this paper presents provides a sketch of a theoretical analysis that could, when it is worked out, help in understanding some of the reasons for success of learning in interactive learning environments with an appropriate set of social activities, such as those that have been discussed by diSessa (1989), by Pea (in press), by Roschelle (1987), and by Schwartz and Yarushalmy (1975). Conceptual learning, according to the ideas that I have presented, is a process that is both socially and materially situated. The goal of learning is an ability to construct and reason with mental models that incorporate conceptual entities and constraints of a domain. An assumption of my discussion is that people learn to do what they do when they learn. The goal of conceptual learning, in my view, is a set of abilities to construct and reason with mental models that incorporate the entities and constraints that constitute the concepts of a domain. To accomplish that goal, we should arrange resources and activities that will cause people to do what we want them to learn—that is, to construct and reason with mental models of the appropriate kinds. In these arrangements, the features of both the physical or computational models and the social activities in which learners interact with the physical or computational models and with each other will play crucial roles.

Acknowledgement: Supported by National Science Foundation Grant MDR-9053605 and the Institute for Research on Learning.

References

Barwise, J. (1989). *The situation in logic.* Stanford, CA: Center for the Study of Language and Information, Stanford University.

diSessa, A. (1989). *Computational media as a foundation for new learning cultures.* Berkeley, CA: Graduate School of Education, University of California, Berkeley.

Devlin, K. (1988). *Logic and information. Volume 1: Situation theory* (draft). Stanford, CA: Center for the Study of Language and Information, Stanford University.

Fauconnier, G. (1985). *Mental spaces: Aspects of meaning construction in natural language.* Cambridge, MA: MIT Press/Bradford Books.

Johnson-Laird, P. N. (1983). *Mental models: Towards a cognitive science of language, inference, and consciousness.* Cambridge, MA: Harvard University Press.

Moore, J. L., & Greeno, J. G. (1991). Implicit understanding of functions in quantitative reasoning. Stanford, CA: School of Education, Stanford University.

Pea, R. (in press). Augmenting the discourse of learning with computer-based learning environments. In E. De Corte, M. Linn, H. Mandl, & L. Verschaffel (Eds.), *Computer-based learning environments and problem solving.*

Piaget, J., Grize, J., Szeminska, A., & Bang, V. (1977). *The psychology and epistemology of functions.* (Originally in French, 1968.)

Roschelle, J. (1990). Designing for conversations. Paper presented at the annual meeting of the American Educational Research Association, Boston, MA.

Schwartz, J., & Yarushalmy, M. (1975). *The geometric supposers.* Sunburst Publications.

Smith, B. C. (1987). The correspondence continuum (Report CSLI-87-71). Stanford, CA: Center for the Study of Language and Information, Stanford University.

Yates, J. (1985). The content of awareness is a model of the world. *Psychological Review,* *92,* 249-284.

A Theory of Manipulatives in Arithmetic Learning

Neil Hall
School of Policy and Technological Studies in Education, University of Wollongong
Wollongong, New South Wales 2500, Australia

Stellan Ohlsson
Learning Research and Development Center
University of Pittsburgh
Pittsburgh, Pennsylvania 15260, U. S. A.

Abstract: Arithmetic procedures are frequently taught with the help of manipulatives, i. e., physical objects which embody numerical concepts. The theoretical rationale for this teaching strategy has never been clearly formulated. Drawing upon current ideas about procedural learning, we propose a theory which predicts the relative effectiveness of different manipulatives. Although manipulatives facilitate procedure acquisition under certain circumstances, they should not be expected to generate conceptual understanding.

Manipulatives in Arithmetic Instruction

A frequently advocated method for teaching children arithmetic is to provide them with physical referents for numerical symbols. A system of objects or quantities which by nature or by design illustrate or instantiate a system of numerical concepts is called a *manipulative* because it allows students to deal with numbers through physical manipulations. Multi-Base Arithmetic Blocks (MBA blocks; also called Dienes Blocks), a system of wooden blocks in different sizes, is a manipulative for place value algorithms. It embodies the idea of place value because each block size differs by a factor ten from the next larger block size and this fact is made visually salient by the design of the blocks. Felt strips that vary in size and color is an alternative manipulative for place value. Pie diagrams and other divided surface areas are frequently used as manipulatives for fractions (Post, 1988).

Manipulatives can be used in different ways in instruction. We focus on a particular teaching strategy which we will refer to as *the three-step scenario:*

(1) *Teach a physical analogue of the target procedure*. The students are taught motor actions (divide circles, exchange blocks, fold papers, move felt strips, etc.) which, within the context of a particular manipulative, correspond to the arithmetic operations carried out by the target procedure.

(2) *Transfer the physical procedure to an intermediate notation.* A notation is introduced which corresponds as closely as possible to the structure of the manipulative, but which may differ from the standard notation. The students are taught to carry out the procedure learned in step 1 with respect to the intermediate notation. For place value algorithms, the intermediate notation is often a so-called *expanded* notation in which the place values of individual digits are represented explicitly.

(3) *Transform the procedure for the intermediate notation into the standard procedure*. Once the procedure is established in the realm of symbols, it is transformed into the standard

> procedure. For example, a procedure for an expanded notation might be transformed into the corresponding standard procedure by deleting the steps in which the place values are made explicit.

Leinhard and Ohlsson (1990) analyzed a videotape of an expert classroom teacher who used felt strips to teach subtraction with regrouping, and found that the teacher followed the three-step scenario quite closely. Little is known about how frequently classroom teachers use this scenario (but see Scott, 1983).

The most puzzling feature of the three-step scenario is that *it multiplies the amount of material to be mastered by at least a factor three*. The student has to learn to operate with the manipulative, the intermediate notation, and the standard notation, so he or she has to learn three procedures rather than one, plus the relations between them. A second puzzling feature is that *the intermediate notation is often more complicated than the standard notation*. To increase the size and complexity of the learner's task are odd ways to facilitate learning. These puzzling features pose the question whether the three-step scenario is, in fact, pedagogically effective. The empirical literature is equivocal; manipulatives are sometimes effective and sometimes not (Fennema, 1972, December; Friedman, 1978; Kieren, 1969; Sowell, 1989).

In order to understand the circumstances under which manipulatives are effective, if any, we need to understand how they function. But the rationale for manipulatives has never been stated clearly. Bruner (1966) hypothesized that manipulatives are useful because learning is a progression through enactive, iconic, and symbolic representations, but this hypothesis is more descriptive than explanatory: It does not tell us *why* the symbolic representation cannot be learned directly, nor does it explain the function of the intermediate notation; why are two symbolic representations better than one? We propose an alternative account of how children learn from manipulatives within the three-step scenario, based on current ideas about procedural learning.

A Theory of Learning From Manipulatives

The three-step scenario can be interpreted in terms of three learning mechanisms described in the literature, namely proceduralization, analogy, and simplification.

Proceduralization**.** The first step in learning from instruction is to encode what the teacher says into a declarative memory structure. In the case of instruction in a cognitive skill, this declarative structure has to be converted into procedural, executable form. Anderson (1989) has proposed that this conversion is carried out by an interpretative problem solving mechanism which accesses the declarative memory structure, generates task relevant actions, and creates procedural rules which can evoke those actions in the future without having to re-access the declarative structure. This is called *proceduralization*. An alternative mechanism for proceduralization has been put forth by Ohlsson & Rees (1991). They suggest that declarative knowledge—including instruction—is encoded in constraints on mental procedures, and that procedures are revised until they are consistent with those constraints. The differences between these two proceduralization mechanisms need not concern us for present purposes. The output from proceduralization is a set of initial rules for the relevant skill.

The first task of a student in the three-step scenario is to proceduralize the instruction he or she receives in step 1. For example, an MBA blocks instruction like "if you need more units blocks, then exchange one tens block for ten units blocks" has to be converted into a procedure for picking up a tens block, moving it to the side, picking up ten units blocks, and so forth. The result is a set of rules for operating with the manipulative. A rough sketch of one such a rule is:

Rule 1: Goal = Get more 1B:s & Has 10B --> Pick up 10B,

where 1B stands for a units block and 10B for a tens block.

Analogy. One strategy for solving an unfamiliar problem is to retrieve from memory some other, already solved problem, and set up an analogical mapping between the two problems. A procedure for the unfamiliar problem can then be constructed by making the appropriate substitutions in the procedure for the familiar problem. The crucial step in analogical transfer is to construct the mapping between the two problems. The differences between the different mapping mechanisms proposed in the literature (Gentner, 1983; Holyoak & Thagard, 1989; Carbonell, 1986; Anderson, 1989) need not concern us here.

In the second step of the three-step scenario, the learner's task is to construct the procedure for the intermediate notation. We suggest that this step is carried out through analogical mapping between the procedures for the manipulative and for the intermediate notation. For example, an analogy between MBA blocks and an expanded notation might include the mappings *Get more <--> Increment*, *1B <--> 1S*, *10B <--> 10S*, *Pick up <--> Cross out*, and so on, where 1S and 10S refer to symbols. Applied to Rule 1, these mappings might generate the following rule for the intermediate notation:

Rule 2: Goal = Increment 1S & Has 10S --> Cross out 10S.

Simplification. Procedural learning progresses from construction to fine tuning. Once a cognitive procedure has been acquired, various simplification processes set in which ultimately lead to automatization. Several different simplification procedures have been proposed in the literature, but for present purposes we focus on a learning mechanism described by Neches (1987) which we will call *redundancy elimination*. The function of redundancy elimination is to delete unnecessary steps from a procedure. Examples of unnecessary steps are steps which produce results which are not used later and steps which yield results which can be produced by a shorter route.

We suggest that the third step of the three-step scenario draws upon the learner's ability to eliminate redundancy. A procedure for an intermediate notation is typically more complicated than the corresponding standard procedure. The process of transforming the former into the latter is thus a matter of simplifying the former.

Implications of the Theory

Why is a detour over a manipulative a more effective route to mastery than direct instruction in the target procedure? One possible answer is that the proceduralization step is easier for children in the case of a physical procedure. Children often have to proceduralize instructions visavi physical objects in daily life (e. g., "hand me the salt, please," "put your socks in the drawer"). An additional advantage is that referring expressions for physical objects, e. g., "the big block," are simple. Proceduralization of a symbolic procedure is likely to be more difficult. The task is unfamiliar and the instructions are likely to contain complicated referring expressions like "the column immediately to the left of the column you are working on"; sometimes the instructions have to refer to entities which have no visible referent, e. g., the place value of a digit. The cognitive cost of proceduralizing the target procedure might be so high that the roundabout route over a manipulative is more effective. *Unless this is the case, there is no reason to expect the three-step scenario to be more effective than direct instruction.*

A second puzzle about the three-step scenario is the function of the intermediate notation. Why does it benefit the learner to have to learn an extra notation? Our theory provides a straightforward answer to this question: The purpose of the intermediate notation is to facilitate analogical transfer. If the standard notation is not isomorphic with the manipulative, an intermediate notation might help bridge the gap between the two. The closer the correspondance between the manipulative and a symbolic notation, the easier it will be for the student to construct the analogical mapping from the physical to the symbolic.

If we assume that the proceduralization and simplification steps are equally complex for different manipulatives, then the main determinant of the pedagogical effectiveness of a manipulative is the degree of isomorphism between the manipulative and the intermediate notation. We have developed a quantitative index of the degree of isomorphism between two procedures. This index measures the number of corresponding goals and actions, and varies between 0 and 1. It yields a low value of .46 for the way in which Resnick and Omansson (19xx) used MBA blocks to teach subtraction, and an even lower value of .27 for the computerized manipulatives used by Ohlsson, Bee, and Zeller (1989) to teach fractions, thus providing a possible explanation for the disappointing outcomes of those two interventions. In comparison, the index value for the use of MBA blocks for teaching addition (Hall, 1982) is .92. In short, our theory predicts the relative effectiveness of different manipulatives on the basis of the degree of isomorphism between the physical and the symbolical. A more complete theory would quantify the proceduralization and simplification steps as well.

A third puzzling feature of the three-step scenario is that the intermediate notation is often more complex than the target notation, a peculiar feature if the former is to help the learner master the latter. But if people are good at simplifying procedures, as research on automatization suggests, then the transformation of the more complex intermediate procedure into the simpler target procedure might impose less cognitive strain than building the target procedure directly.

Summary and Discussion

The rationale for the use of manipulatives in arithmetic instruction has never been stated clearly. How can manipulatives and intermediate notations facilitate learning, even though they require the learner to master three procedures rather than one? Our theory claims that the pedagogical effectiveness of a manipulative is a function of three variables: (a) the cognitive complexity of proceduralizing the instruction for the manipulative, (b) the obviousness of the analogical mapping between the manipulative and the intermediate notation, and (c) the cognitive complexity of transforming the intermediate into the standard procedure.

According to our theory, manipulatives will be more effective than direct teaching *only* in those cases in which the cognitive cost of proceduralizing the standard procedure is higher than the summed cognitive costs of the three-step scenario. There are many reasons to expect the proceduralization of a symbolic procedure to be difficult for young children, so it is plausible that manipulatives are more effective than direct instruction in some cases. A detailed prediction about particular cases would require quantification of the cost associated with proceduralization, a task for future research.

Our theory also makes predictions about the relative advantage of different manipulatives. If we assume that the cognitive complexity of the proceduralization and simplification steps are approximately the same for different manipulatives, then the obviousness of the analogical mapping between a manipulative and the intermediate notation is the main determinant of the effectiveness of the manipulative.

However, contrary to the current *Zeitgeist* in mathematics education, our theory provides no support for the widespread idea that manipulatives improve students' conceptual understanding of arithmetic. The key hypothesis in our theory is that students create a symbolic procedure through analogical transfer from the physical procedure for operating on the manipulative. There is no reason to expect such transfer to generate understanding of the mathematical rationale for the symbolic procedure.

References

Anderson, J. R. (1989). A theory of the origins of human knowledge. *Artificial Intelligence, 40*, 313-351.

Bruner, J. S. (1966). Notes on a theory of instruction. Bruner, J. S. *Toward a theory of instruction* (pp. 39-72). Cambridge, MA: Harvard University Press.

Carbonell, J. G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R. S. Michalski, J. G. Carbonell, & T. M. Mitchell, (Eds.), *Machine learning. An artificial intelligence approach. Vol 2* (pp. 371-392). Los Altos, CA: Kaufmann.

Fennema, E. H. (1972, December). Models and mathematics*Arithmetic Teacher*, 635-639.

Friedman, M. (1978), The manipulative materials strategy: The latest pied piper? *Journal for Research in Mathematics Education, 9*(2), 78-80.

Gentner, D. (1983). Structure-mapping: A theoretical framework for analogy. *Cognitive Science, 7*(2), 155-170.

Hall, N. (1982). Towards a theory of teaching mathematics. In J. Veness, (Ed.), *Mathematics - A universal language* (pp. 10-25). Sydney, Australia: AAMT.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. *Cognitive Science, 13*, 295-355.

Kieren, T. E. (1969). Activity learning. *Review of Educational Research, 39*(4), 509-521.

Leinhardt, G. & Ohlsson, S (1990). Tutorials on the structure of tutoring from teachers. *Journal of Artificial Intelligence in Education, 2*, 21-46.

Neches, R. (1987). Learning through incremental refinement of procedures. In D. Klahr, P. Langley, & R. Neches, (Eds.), *Production system models of learning and development* (pp. 163-219). Cambridge, MA: MIT Press.

Ohlsson, S., Bee, N., & Zeller, P. (1989). *Empirical evaluation of a computer-based learning environment for fractions* (Report #KUL-89-07). Pittsburgh, PA: Univ of Pittsburgh.

Ohlsson, S., & Rees, E. (1991). Adaptive search through constraint violation. *Journal of Theoretical and Artificial Intelligence*, 3, 33-42.

Post, T. R., (Ed.), (1988). *Teaching mathematics in grades K-8. Research based methods*. Boston, MA: Allyn and Bacon.

Resnick, L. B., & Omanson, S. F. (1987). Learning to understand arithmetic. In R. Glaser, (Ed.), *Advances in instructional psychology* (Vol 3, pp. 41-95). Hillsdale, NJ: Erlbaum.

Scott, P. B. (1983). A survey of perceived use of mathematics materials by elementary teachers in a large urban school district. *School Science Math, 83*(1), 61-68.

Sowell, E. J. (1989). Effects of manipulative materials in mathematics education. *Journal for Research in Mathematics Eduation, 20*(5), 498-505.

Qualitative Modelling and Conceptual Change in Science Students

J.R. Hartley, M.J. Byard, and C.L. Mallen
Computer Based Learning Unit, School of Education
University of Leeds, Leeds, UK, LS2 9JT

Abstract: It is well documented that students' interpretations of everyday experience can lead to naive views of (Newtonian) Physics and to the development of alternative conceptual frameworks: how to deal with such misconceived but comfortable sets of beliefs is much less clear-cut. Our research has concentrated on students' understanding of motion with frictional and gravitational forces, and two types of supporting software have been designed and implemented. The first (DM3) uses simulation techniques with a Newtonian model, and a direct manipulation interface in which icons are linked to well known contexts. The aim is for students to investigate and learn the operational rules governing motion in these environments. Following this experience, the second software package (VARILAB) allows students to build explanations as qualitative models, where agent-effects on object-attributes govern state-changes of the system and hence its animation and graphical outputs. The research hypotheses concern the ways beliefs may be changed through the detection and resolution of inconsistencies between cause-effect properties and the operational rules that seem to govern events. Studies with pairs of fourteen to sixteen year old students are focusing on action and dialogue patterns to yield descriptions of the processes of (or resistance to) conceptual change.

Students' Conceptions and Explanations in Science

Science, whether formal or naive, is an interpretative system which we overlay on observations and experience in the natural world so that we can better understand (and exploit) it. Expert science uses the agreed vocabulary and conventions of the acknowledged science community; naive views are less formalised and ambitious in their purposes, but refer to the more-or-less organised sets of beliefs which the wider community use in explaining events in the physical world. It is well-accepted that the prior experience and explanatory frameworks which students bring to learning tasks are a major influence in the development of understanding. However, when these frameworks are not consistent with more expert views, then learning difficulties can arise perhaps resulting in a compartmentalisation of formal science knowledge and/or leading to uncertainties in conceptual knowledge. The problem can be further complicated since the language used by the science teacher (e.g., force, energy) may be translated in a looser informal sense that misses or ignores the precision being introduced. This is likely to be the case if the representational techniques of science, such as a graphical or algebraic methods, are themselves not well understood by students.

Before considering ways of dealing with these teaching/learning problems, it is worthwhile making brief comments on students' alternative conceptions and how they have been examined and interpreted by researchers. At Leeds, as part of the Conceptual Change in Science Project, a study was undertaken with twenty pairs of students, in the age range ten to fifteen years, from three local schools. The contexts of interest were taken from physics and featured horizontal and vertical motion, including movement of a toy railway carriage, pulling a dynamics cart, cycling along level surfaces, throwing a ball, and the motion of a parachutist. Typically, students were shown an event (e.g., a push given to the model railway carriage), and investigator-led questioning invited a description of the event, causal explanations, and discussion of what-if conditions or consequences, (e.g., if the push were increased or another carriage attached). All interviews were video-recorded.

Details of the results are available (Twigger and Byard, 1989) but our purpose here is to comment on the nature of the misconceptions rather than to provide a comprehensive listing of their content and expression. By mis- (or alternative) conception is meant a held belief that is defended in argument and used consistently in explanations across contexts. Hence, it is not a mistake arising from ignorance, and corrected by feedback, neither is it a confusion or uncertainty, for example, between mass and weight. The principal example of erroneous conceptions, expressed in many forms in the experimental study, is that force is needed to maintain motion, and with no (net) force there will be no motion. That this is a misconception, rather than an observation of experience incompletely or carelessly expressed, is shown by questions asking about changes in speed; for example, when the force is removed. A student may reply that a surrogate force is envisioned in the body (perhaps called force/energy), thus bringing a cause close to its effect, and this 'force' is then used up (decreased) so driving the body at a slower speed.

For such conceptions, consistency is often maintained or defended across contexts (e.g., gravity is a constant force so bodies fall at a constant speed) and held even when contradicted by demonstration (e.g., pulling and accelerating the cart with elastic at constant stretch can bring comments about peculiar properties of elastic, ruling it out as an exception). Other arguments are more sophisticated seeing the motion of a ball thrown vertically as a 'combat' between the surrogate-force and gravity; gravity wins at the top of the arc, "pulling the ball down" at a greater speed and in a shorter time than that of its ascent. Others recognise symmetry in the upward-downward motion since the 'force' is judged to be equal to gravity at the top of the arc, and only becomes zero when the ball returns to the ground. In summary, such misconceptions are integral to students' explanations, are held consistently and protected by concentrating on the system-body state rather than on the processes of change, by ignoring boundary conditions ("the force and friction may be equal but the body goes from left to right so the force is just a little bit more"), by context peculiarities (e.g., properties of elastic), and by misperceived "facts" (e.g., under gravity-force bodies fall at constant speed).

The Leeds Study, in its range of alternative conceptions, supported, for example, previous research of Viennot (1979), Clement (1980), McDermott (1984), and Driver (1984). Such intuitive notions were held by students with differing ages, experiences and abilities; it is worth noting also that there was uncertainty about other related concepts such as friction, inertial mass, gravity force/acceleration and, for the younger students, some difficulties about representing event descriptions in a graphical form. Thus, tackling problems of alternative frameworks must relate to students' knowledge (or knowledge-lack) of the complex of concepts involved in motion.

Interpreting Alternative Conceptions

Various accounts and interpretations have been given of alternative conceptions. Constructivists, in various guises, seek to represent successive stages of science understanding as changing patterns of variables, seen to be important in events, and their interrelations. This permits varying mental constructions to provide explanations that are shaped or changed through differences between observed effects and expectations (Driver, 1983). Conceptual change thus emerges from a process of controlled investigation and enquiry.

This process of students' scientific reasoning, which is qualitative in character, has been examined by Forbus (1984), and deKleer and Brown (1984). The latter distinguish between an event description space, containing the interacting components (i.e., the structure) of the physical system, and the causal account that comes from working out and running a mental model of how such a system might work. If this is expressed as a computer program, then the results can be compared with 'reality'. Forbus, in his approach, emphasizes the (qualitative) processes that bring about the state-changes in the system, and application across different contexts can lead to the development of generalised models. Experimental evidence, however, emphasizes that students, in building explanatory models, use intuitive notions, though in a systematic way. diSessa (1989) considers these knowledge elements to be "phenomenological primitives - p-prims" that form a priority structure so that some are recognized as being more relevant to a particular context than others. These intuitions of "how the world works" loosely interlink to maintain a collective justification that is much more than the strength of belief in an individual p-prim (diSessa and Globerson, 1984). This interpretation relates well with theories of belief systems (e.g., Harman, 1986) who points out that beliefs are not linked in a hierarchy with justification coming from deeper or more fundamental beliefs. Indeed, we do not keep mental retrievable records of reasons for various beliefs; if pressed, we construct some evidence, perhaps an example from experience. In short, beliefs are roughly all-or-nothing units that can be employed to provide explanations.

The main conclusion is that belief-revision acts under conservative policies: if there is no good reason to change beliefs, we let them rest. Indeed, Posner, et al. (1982) cite four conditions that must be met before students accommodate changes to their existing belief frameworks. These are a dissatisfaction with existing conceptions and meeting new conceptions that are intelligible, plausible when matched to experience, and have added value in suggesting a "fruitful 'research' programme." This inertia of belief will apply particularly if concepts are ambiguously expressed as in naive physics (where energy, force, and power are often given similar interpretations), or when they can take on forms to suit particular types of contexts. The major difficulty with the conservative properties of belief is that they act as constraints on thinking. Why bother to look at things from another viewpoint if one is satisfied with current beliefs?

Changing Students' Conceptions

How then can revision of beliefs be brought about? A noted tendency, well-discussed by Harman (1986), is to seek coherence. On the one hand, there is a tendency to avoid or resolve inconsistencies of belief and, on the other hand, to relate or make interconnections between them, for example, by implication or by developing explanatory systems. Both these activities involve comparing and contrasting event descriptions and running causal explanations against experience (plausibility checks). Hence, the descriptions of physical systems and events have to be given a proper representation and structure in terms, perhaps, of key variables and processes that can allow associated mappings, not only by direct matching of components and effects but by implication and analogy. Thus, we not only require a language and representation for descriptions, but one that can be made operational for explanation and 'experiment'. Harman also notes that beliefs stimulate practical reasoning involving plans and actions. In science, it is these actions which cause the effects and side-effects from which new and changed knowledge can arise. However, beliefs are only likely to be examined, related to action, put to test, and changed if they are felt to matter, and this emphasizes the motivational aspects of learning.

Thus, several researchers have pointed to the advantages of using computer based simulations and modelling systems to influence conceptual change. Of related interest is the place of dialogue between participants using such systems as a student group (where they have to reconcile, coordinate, or override differences in beliefs as they agree on their action sequences) or with the teacher. For example, Brna (1989) advocates the use of modelling methods to expose, and confront directly, misconceptions. Brown and Clement note that conceptual gains only became significant when increased emphasis was placed on students' verbalizing and discussing their qualitative explanations. They also recommend systems that permit "finer detail of the conceptual territory," and are modular in design so that they can be re-visited for frequent comparison.

Software to Engineer Conceptual Change

To develop and test these views of conceptual development and change, our project (which is a collaboration between researchers at Leeds and Glasgow and the Open University) designed and implemented two software systems. The first (from partners at Glasgow and the Open University) was a simulation package in which users could investigate and learn the operational 'rules' of a physics-motion 'world'. The simulation uses a Newtonian model on which contexts are placed through cartoon-icons. The system design is object-based so that attributes of the world (such as gravitation), and their interactions with varieties of objects that can move in the world, are easily managed: at each time-step all forces or other agents linked to objects are activated and new velocity, position and other affected attribute values are calculated.

The motive forces themselves are directly manipulated by the user (e.g., through sliders and/or buttons) and their effects can be shown as a visual animation of events through time, with dial-values, ticker-tape tracks, and graphical output. Recording- and replay buttons enabled users to review these action-event sequences. An example for a horizontal-motion and friction context is shown in Figure 1, where, through the force-slider, the person-icon can give different strengths-of-push to apply to the box (which can contain different masses) at any time. These events can happen on the polished floor of the supermarket, or on the rougher higher-level friction surface of the car park. The system (labelled DM3) incorporates a Lesson Office to store, tune and window manage the various contexts and their lessons, and has been described elsewhere (Twigger et al., 1990).

For our purposes DM3 has several design features of interest. First, the prepared contexts are idealised (but well-known) and make ready links to experience, with cartoon-icons giving representations of everyday situations. To-date these contexts include horizontal motion with and without friction (pushing boxes in supermarkets and skating), and motion against a variable resistance (a speedboat scenario). For vertical motion there was parachute and free fall. This sequencing of contexts allows complexity to be added in

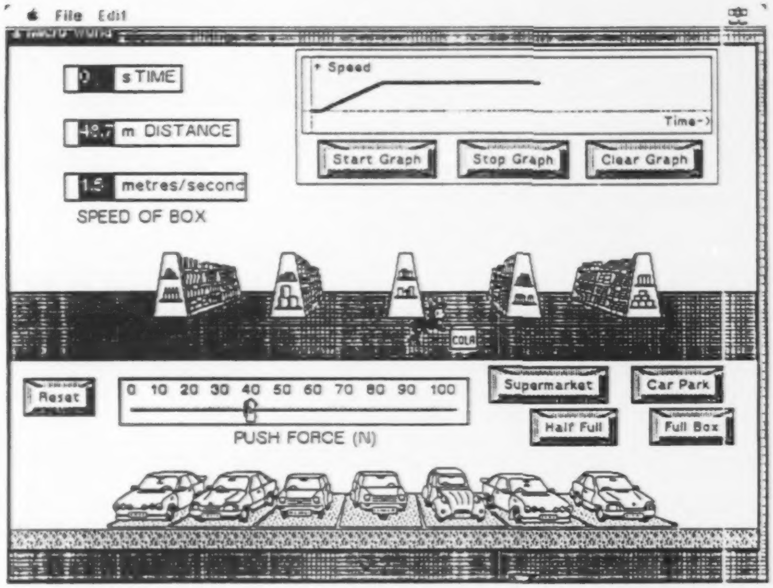


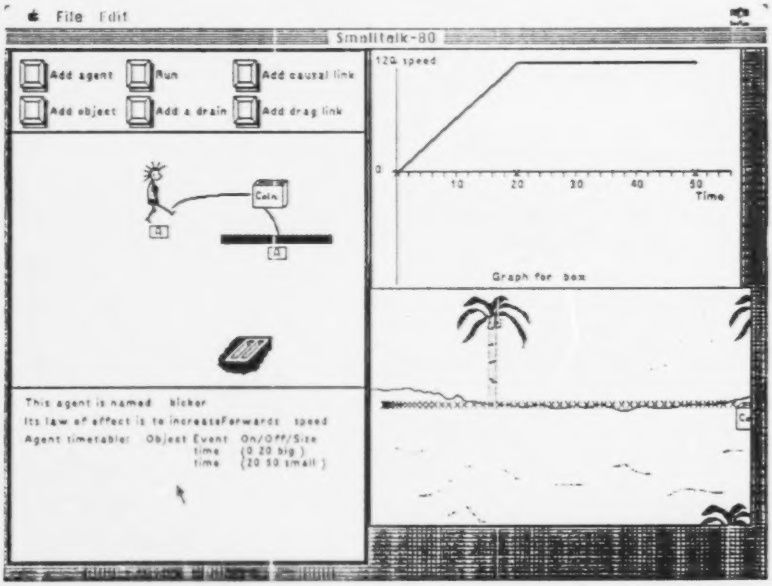
Figure 1. An Example of DM3 Interface

ways that are controlled and which invite comparison between operational rules that can manage motion in the various scenarios. A second feature is that the user can directly manipulate the operation of the scenario and the force effects through slider controls at any time, and the effects of these actions are seen directly and immediately through the animation and the graphical displays. Also the object-oriented design allows realities of friction or gravity (say) to be exaggerated or given an alternative functional representation.

The second and related software system, VARILAB, is a computer based language through which the user can describe a physical system, and express (qualitative) ideas about cause-effect relations that govern its operation. From these expressions, the computer system tries to build a manifestation that will run, i.e., exhibit some autonomous behaviour which can be compared with expectations, or with real-life criterion behaviour. This may lead the user to a correction, or restructuring of concepts to make them more accurate and/or consistent. The operational 'rules' forming such explanations may have been gained from DM3 experience and, in VARILAB descriptions, the DM3 icons rather than more general ones may be used. In fact, the developed explanatory model can serve as a replacement for the DM3 simulation engine.

In essence then, through VARILAB, the user and the computing system maintain a dialogue. The former tells about object components, attributes and structure, together with causal rules as to how changes and effects are brought about. The system shows its understanding of what it has been told, applies its generic knowledge of object agent systems in constructing the explanatory model, in showing consequences and in reminding, preventing or noting odd features about the information it has been given. For these interchanges a simple 'language' had to be designed capable of accommodating the users knowledge and requirements yet linking to the generic and formalised notions held by the system [cf. Vygotsky's (1978) notions of distributed information among participants, and language mechanisms that permit the transmission and understanding of such information]. In short, the language should be usable by students and support their expressions of ideas and intuitive reasoning or alternative conceptions, as well as being powerful enough to permit the modelling of physical systems. Its aim is to externalize the students' reasoning and thus bring it into the common ground so that, if necessary, teachers can engineer further discussion that encourages students to move to more accepted views.

The VARILAB language has its central constructs concerned with notions of cause/effect or agency, and is loosely related to Qualitative Process Theory (QPT) developed by Forbus (1984) with a touch of the envisioning theory of deKleer and Brown (1984). The behaviour of a system is described through state-changes that objects within the system undergo through time. In general, for the state of an object to have changed, an agent must have acted on it [note that the distinction between object as 'receiver' and agent 'as initiator' describes only their functional roles within a given scenario]. Thus, objects are described by a set of attributes (e.g., mass, speed, colour) and their associated 'values'; agents bring about changes to these states by a 'law of effect'. These object-agent connections are specified through drawing a (causal) link and specifying the attribute affected by a monotonically increasing or decreasing relation (e.g., speed, increase). To complete the description, a size of effect (from very small to very big) has to be included (also direction can be introduced) and the activity period of the agent specified. This can be done on the basis of time or when an object reaches (or is in) some specified state. The output of the runnable model can be shown as an animated ticker-tape or in graphical form for the attributes specified by the user. A simple example of a VARILAB model is shown in Figure 2.



```
This agent is named   kicker
Its law of effect is to increaseForwards   speed
Agent timetable:   Object Event   On/Off/Size
                          time    (0 20 big )
                          time    (20 50 small )
```

Figure 2. A Simple Illustration of a VARILAB Model

Two further points should be noted. As well as using causal links, the user may also employ 'drain' and 'drag', envisioning links to allow reactive forces–such as friction–to be differently expressed. A 'drain', for example, decreases the attribute of an object (e.g., its speed) to zero, and a 'drag' acts similarly except that a link has to be made to the object (e.g., the floor) that is implicated. A second VARILAB feature or constraint is that one agent has to be responsible for one effect. However, attributes of an object, e.g., mass and speed, can be inter-related directly or inversely, thus defining a 'composite' (e.g., momentum) to which the agent can then be linked.

The specific terms used for the object-agent descriptions are supplied by the student though, as noted previously, icons from the DM3 library can be employed. On its part, the system has notions of consistency, i.e., an agent cannot vary its law of effect but can regulate its magnitude, and the 'law of activity' ensures that an agent can only affect an object during its 'ON-state'.

During several months both the DM3 and VARILAB software was extensively trialled in schools, particularly to guide the interface design and to evaluate the preliminary training materials (Twigger et al., 1991).

A Study of Conceptual Change

During January to April 1991, a laboratory study was undertaken with ten pairs of students, aged fifteen years, taken from local comprehensive schools. The aims were to note instances of conceptual change/no-change during and following a sequence of study tasks based on the software, and to examine these processes of change. Each pair of students worked in a small room, in the presence of an experimenter who followed strict rules for intervention and discourse; the student-pair were encouraged to talk as they worked, and all sessions were videotaped. The sequence of chosen contexts for the study tasks were focused on horizontal motion with and without friction (using supermarket, car park and skater scenarios) and vertical motion (throwing a ball). With pre- and post-tests (including both pencil-paper items and discussion using other similar scenarios), and training sessions for using the software (which took about one hour), each pair of students required approximately seven hours in total to complete the experiment.

For each scenario, the session was organised as: (i) Description of the context and types of events followed by experimenter-questioning of effects and causes in order to establish misconceptions and knowledge base-lines; (ii) Worksheet of tasks on the DM3 simulation in which students were asked to investigate motion, and also consider the operational rules governing motion, i.e., getting the body started, going faster, slowing down/stopping and moving at constant speed; this was followed by probe questions/discussion from the experimenter; and (iii) Use of the VARILAB software to build an explanatory model that could replicate the DM3 output. During this part of the study, the experimenter acted as a minimal aid in case of difficulties with the software, or if students reached an impasse in their reasoning. When the model had been produced, the experimenter was allowed to ask questions (on features of the model, implications, and comparisons to other models) that required commitment from the student. Also, the experimenter could challenge the students, thus requiring an explanation, or point to inconsistencies that the students should resolve. Where the model was in contradiction to the expressed beliefs, the 'confrontation' was made clear.

Some Concluding Comments

The experiment ended in March 1991 and, at this stage (with the analysis uncompleted), only preliminary comments can be ventured. The data is being examined at three levels. First, there is the conception/misconception history shown through the performance differences between the pre-tests, the stage-probes and post-test data. Second, there is the detail of the process of conceptual change, achieved through analysis of work patterns with the software, and with the associated comments of the student pairs. Finally, there is discussion of particular areas of interest in individual case studies.

All students showed misconceptions, principally those associated with net-force being a requirement for constant motion, and a surrogate-force being envisioned within a body so that the origin-force is given a 'direct' effect after it has ceased to act. The study sessions seemed able to correct the large majority of these conceptions, judged on the basis of answers to probes, and on post-test discussion using other scenarios. Some of these changes were at the DM3 stage but, as expected, most occurred following VARILAB. However, for some students there were re-occurrences of misconceptions in other (later) scenarios and in the post-tests, though these were quickly corrected following an experimenter check. Nevertheless, there were occasions, sometimes with more able students, when confrontation between constructed models and alternative conceptions could not shift 'psychological' belief.

In examining the process of change, students' comments were analysed at the propositional level into descriptive observations, causal inferences, assertions of world behaviour, and instructions. Although DM3 was used for operational tasks, the largest category of comments was that of causal inference, showing organisational thinking. In building VARILAB models, no group used the drain/drag links (even though these corresponded most closely to their intuitive explanations); instead, students preferred the control of active causal links. There was strong carry-over from DM3 experience but, in both cases and particularly in VARILAB, the experimenter's question and discussion were important. For example, VARILAB tasks could become specificactions to produce a certain graphical output, and comments (and attention) could ignore the characteristics and properties of the model itself. Building a model does not ensure it is understood. Also, most of the comments in DM3 and VARILAB task stages were specific, and few generalisations and comparisons to previous experience were made. Bringing a wider focus of attention, as well as discussion of causal account/VARILAB differences, were the main substance of the experimenter's dialogue with students.

The individual case studies showed useful learning of a range of concepts including friction properties, mass/weight distinctions and gravity force/acceleration. [For example, the DM3 scenarios showed that heavier and lighter objects slowed at the same rate under linear friction when the pushing force was removed, and the VARILAB model required a proportionate adjustment in the friction force levels.] Perhaps unexpectedly, the data showed that less experienced students (i.e., those who had less understanding of these associated concepts) had more misconceptions, but these proved easier to remove than for more experienced students who were better able to defend their alternative viewpoints. Also, there were several examples where misconceptions acted as a severe block to learning. One pair, for example, in the vertical throw of the ball, were convinced that, at the top of the flight with zero velocity, the gravity force and the surrogate throw-force within the ball were equal. In modelling this situation in VARILAB, the throw-force was progressively reduced against a constant gravity which eventually levelled off the speed at a constant value.

Since they were unshaken in their belief of the equilibrium condition, they reached an impasse. As a final note, even when the most able student-pair had produced and understood a correct model but one which was at variance with their intuitive explanation, a participant still could not "believe" the VARILAB view. Maybe this psychological aspect is not always important, provided students can operate the Newtonian model and know when it is appropriate to do so. Perhaps by this stage intuitive models have outlived their usefulness.

Acknowledgements: The authors readily acknowledge the work and help of their partners in the Conceptual Change in Science Project, supported (under the InTER Programme) by the Economic and Social Research Council, UK: in particular, we note Daz Twigger and Rosalind Driver at Leeds, Stephen Draper at Glasgow, and Tim O'Shea, Eileen Scanlon, Sara Hennessy and Ramzan Mohamed at the Open University. Also, we are grateful for the ready participation of the Leeds schools and students in the experimental studies.

References

Brna, P. (1989). Programmed rockets: an analysis of students' strategies. *British Journal of Educational Technology* 20(1): 27-40.

Clement, J. (1982). Students' preconceptions in introductory mechanics. *American Journal of Physics* 50(1): 66-71.

deKleer, J. and Brown, J.S. (1984). Qualitative physics based on influences. *Artificial Intelligence* 24: 7-83.

diSessa, A. (1989). Knowledge in pieces. In G Forman, and P.B. Pufall (eds), *Constructivism in the computer age,* pp. 49-70. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

diSessa, A. and Globerson, T. (1984). The effects of age and cognitive style on children's intuitions of motion. Paper presented at LOGO 84, Cambridge, MA, U.S.A.

Driver, R. (1983). *The pupil as scientist?* Open University Press, U.K.

Forbus, K. (1984). Qualitative process theory. *Artificial Intelligence* 24: 85-168. Harman, G. (1986). Change in view: principles of reasoning. MIT Press. McDermott, L.C. (1984) Research on conceptual understanding in mechanics. *Physics Today* 37: 24-32.

Posner, G. J., Strike, K.A., Henson, P.W., and Gertzog, W.A. (1982). Accommodation of a scientific conception: towards a theory of conceptual change. *Science Education* 66(2): 211-227.

Twigger, D., Byard, M.J., Draper, S., Driver, R., Hartley, J.R., Hennessy, S., Mallen, C.L., Mohamed, R., O'Malley, C., O'Shea, T., and Scanlon, E. (1991). *Journal of Computer Assisted Learning.* In press.

Twigger, D. and Byard, M.J. (1989). A preliminary study of alternative conceptions in science by schoolchildren. Internal Report, Centre for Studies in Science Education, University of Leeds, Leeds, U.K., LS2 9JT.

Viennot, L. (1979). Spontaneous reasoning in elementary dynamics. *European Journal of Science Education* 1(2): 205-221.

Vygotsky, LS. (1978). The development of the higher psychological processes. *Mind in Society.* Cambridge, MA, U.S.A. Harvard University Press.

Guided Discovery Learning: The Computer as a Cooperative Learning Assistant

David Hintze
Computer-Based Learning Unit,
University of Leeds, Leeds, U. K., LS2 9JT

Abstract: The aim is for the student and a learning-assistant program to engage in a discovery learning exercise which is genuinely cooperative. Although having this generic formulation, the working context is the British Telecom Merlin telecommunication system simulated on a Sun3 workstation. The learning-assistant program has no expert knowledge of the domain. The student uses a work record to segment and label sequences of key presses which are judged to be useful. The program knows how to propose and test generalisations (including those suggested by the student) against this work record and thus is able to build up an "understanding" of the functionality of the Merlin commands and the tasks which can be accomplished with the communication system. The prototype assistant was built following experimental studies with pairs of learners.

Introduction

Within computer-based learning there have been notable projects in learner-control systems eg TICCIT (Merrill, 1980) and in the use of simulation techniques. However, there are difficulties in realising the potential of such schemes. In the TICCIT system, learners did not have enough advisory and decision support, so that attrition rates were large (Alderman, 1978). Simulations can also encounter problems if students do not have adequate preparation and sufficient conceptual understanding to guide their hypothesis formulation and testing in a principled manner (Bork and Robson, 1972). Hence it is accepted that such open and individualised learning systems should be a cooperative process of "guided discovery." In practice though, the methods of the human tutor in the classroom or hints within computer-based programs operate from the expert viewpoint. The learner is guided, more or less, rather than allowing the students to exploit their own knowledge bases and viewpoints. In this research we envisage the student and the co-learning program undertaking a joint cooperative learning activity. The co-learning program has no expert knowledge of the domain, but does have the general ability to keep state-change records, engage in generalisations and consistency checks and undertake elementary planning. We envisage communication between the student and the learning program about system state-changes, functional effects and task goals. The student has to have a means of stating and "explaining" these terms to the co-learner program so they can share common vocabularies and representational frameworks.

The working context was a procedural system (the Merlin communication system) and to allow all the collaborative working to take place on one workstation, a Merlin simulator was built. This was to serve also as an experimental tool for studying the techniques followed by human learners when engaged on a cooperative learning task.

The Merlin console consists of a keyboard, a display unit and a telephone handset (Figure 1).

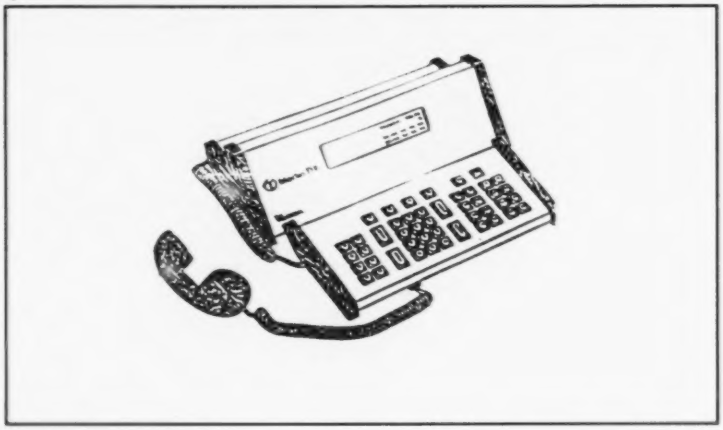


Figure 1. The Merlin console

The system is controlled by pressing keys on the keyboard. The state of the system is shown by the appearance of brief messages on the display and by the illumination of the source (S) keys and the loop (L) keys.

In the simulation (Figure 2), keys are pressed by moving the mouse cursor over a key and pressing a button on the mouse controller. The vocal input is simulated partly by messages appearing in one of the windows and partly by asking the subjects to play both the roles of the operator and of callers. This means it is up to the subjects whether a call is answered or left ringing, or when a call is terminated. The subjects are able to examine and control the state of various aspects of the system. For example, by selecting "Ringing Extensions," a list of the extensions which are currently ringing appears. By selecting one of the extensions, a pop-up menu appears which offers the subjects the option of answering the extension, timing out the ringing extension and so returning the caller to the console or allowing the extension to continue ringing.

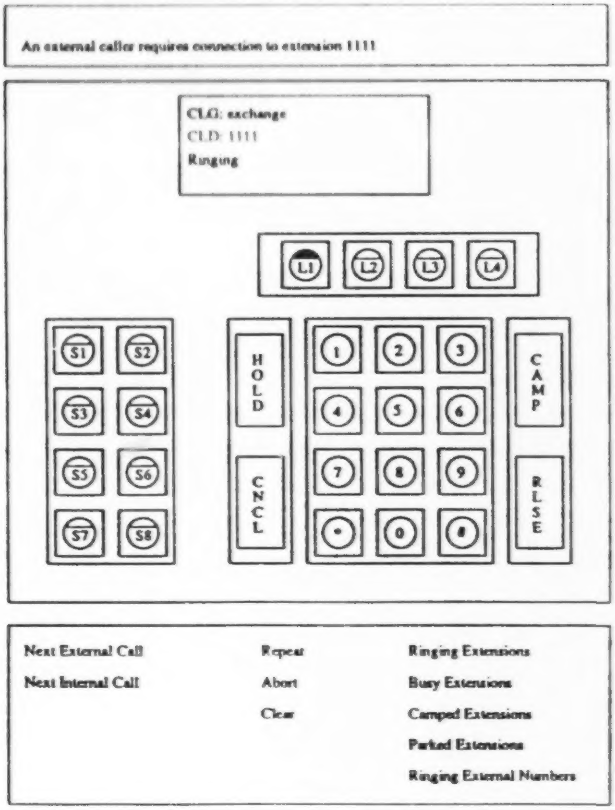
Conduct of the Experiments

The study was to examine the actions, explanations, suggestions and discussions which engaged human subjects as they undertook the cooperative learning task of discovering how to operate Merlin; a principal objective was to guide the requirements and design of a proposed co-learner program.

Five pairs of subjects were given two one-hour sessions with the Merlin simulation. Sessions were separated by several days. Each pair was given a brief written description of the aims and method of the study.

The instructions emphasised that the subjects were to discuss aloud their ideas and observations. The conversation was recorded. Each key pressed by the subject was also recorded, associated with a time stamp that would allow the conversation and key presses to be integrated into a written protocol.

The subjects were allowed to progress through the experiment without interference from the experimenter. The experimenter did intervene, however, to give some minimal advice if subjects were unable to proceed.



An external caller requires connection to extension 1111

CLG: exchange
CLD: 1111
Ringing

L1
L2
L3
L4
S1
S2
S3
S4
S5
S6
S7
S8
HOLD
CNCL
1
2
3
4
5
6
7
8
9
*
0
#
CAMP
RLSE
Next External Call
Next Internal Call
Repeat
Abort
Clear
Ringing Extensions
Busy Extensions
Camped Extensions
Parked Extensions
Ringing External Numbers

Figure 2. The Merlin simulation showing the state of the system after the operator has rung extension 1111 in order to put through an external caller.

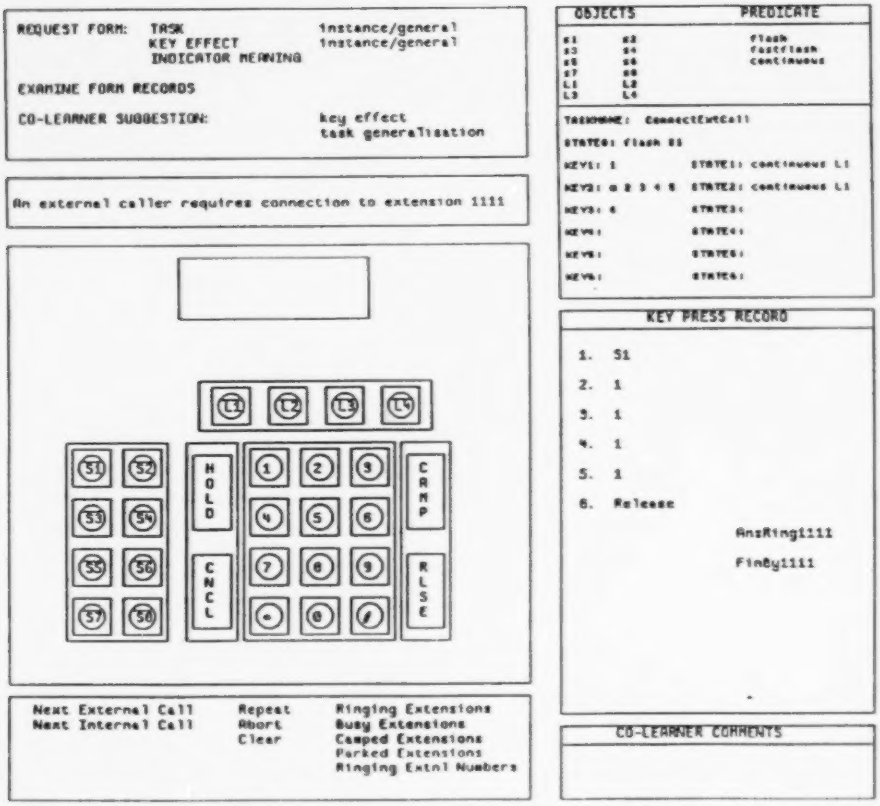
Results

The recorded conversations from the experiments were transcribed and analysed.

First, it was noted that discussions between the subjects centred on three aspects of Merlin:

(1) How to achieve task-goals; e.g., Connecting an external call to an extension.
(2) The function of individual keys and groups of keys; e.g., What do the loop keys do?
(3) The significance of the indicator states and what events cause their state changes; e.g., what does a flashing loop key mean?

A second feature of the learning data showed that subjects found it difficult to remember the sequence of key presses which they had used to arrive at a state which they considered to be significant.



```
REQUEST FORM:  TASK                    instance/general
               KEY EFFECT              instance/general
               INDICATOR MEANING

EXAMINE FORM RECORDS

CO-LEARNER SUGGESTION:                 key effect
                                       task generalisation
```

```
An external caller requires connection to extension 1111
```

```
OBJECTS                          PREDICATE
S1        S2                     flash
S3        S4                     fastflash
S5        S6                     continuous
S7        S8
L1        L2
L3        L4

TASKNAME:  ConnectExtCall

STATE0: flash S1

KEY1: 1              STATE1: continuous L1

KEY2: 0 2 3 4 5      STATE2: continuous L1

KEY3: 6              STATE3:

KEY4:                STATE4:

KEY5:                STATE5:

KEY6:                STATE6:
```

```
KEY PRESS RECORD

1.  S1

2.  1

3.  1

4.  1

5.  1

6.  Release

                AnsRing1111

                FinBy1111
```

L1
L2
L3
L4
S1
S2
S3
S4
S5
S6
S7
S8
HOLD
CNCL
1
2
3
4
5
6
7
8
9
*
0
#
CAMP
RLSE
Next External Call
Next Internal Call
Repeat
Abort
Clear
Ringing Extensions
Busy Extensions
Camped Extensions
Parked Extensions
Ringing Extnl Numbers
CO-LEARNER COMMENTS

Figure 3. The Merlin simulation with the additional co-learner windows. The student has successfully connection an external caller to extension 1111 and is now filling in a task instance form.

Thirdly, from particular observations the subjects were able to construct and test hypotheses at a more general level. For example, from observation of the effects of pressing the "Release" key in specific instances, the subjects formulated ideas about its use in general. Hence generalisation and its use in the construction and testing of method plans was an important technique in learning.

Preliminary Conclusions

Our aim in carrying out the experiments was to indicate where a co-learner program could be of assistance to a human participant in a discovery-learning situation. From these data, the design of the co-learner has concentrated on:

(1) Providing an aid to memory in the form of a record of all key presses made by the student.
(2) A notepad in which the student can record observations and hypotheses. The co-learner also has access to the notepad, so it is able to criticise, and/or generalise and check (i.e., test for consistency against the key press record) the student's ideas in a constructive way.

The notepad is the common medium by which the student and co-learner communicate their knowledge.

(3) Allowing the co-learner to come up with assertions and interpretations (at a higher level of generality) based mainly on what the student has recorded in the notepad, but with some information from its own records. In keeping with the notion of a co-learner rather than an expert tutor, the co-learner's ideas are not guaranteed to be correct and are open to criticism by the student.

Design of the Co-learner

We have constructed a prototype of a co-learner consisting of some additional windows which are displayed concurrently with the Merlin simulation on a Sun3 workstation (Figure 3).

The additional windows include a key press record which is permanently displayed and contains a list of the key presses made by the student. Both the key presses made in the role of operator and those made in the role of caller are recorded and the Merlin key presses are numbered so they can be referred to when task instances are added to the notepad. The student can browse the list as a reminder of which keys were pressed to arrive at a particular state.

The co-learner program also has access to the key press record, but in addition to the key presses, it keeps a record of the state of the indicators before and after each key press.

A notepad pop-up window also appears at the request of the student and contains a form which the student completes using a simple representation language in order to segment and describe the work record into tasks and sub-tasks, and to suggest assertions and generalisations. The terms in the language which are applicable to that form appear in a window alongside the pop-up window.

The Representation Language and Interactions

The language is a simplified form of predicate calculus and is used directly by the student (aided by the experimenter if necessary at present) and by the co-learner program. The language is used to describe the state of the indicators and the state of the calls on the exchange: it contains variables, which are available to use in forming generalisations.

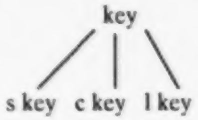
Examples:

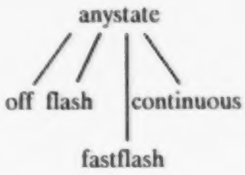
flash (S1) means "S1 key is flashing"

flash (x) * skey (x)
means "an unspecified S key is flashing"

An asterisk signifies a conjunction.

The co-learner is able to deal with two types of generalisation. The first uses a taxonomy of predicates, though there are only two taxonomies at present.





For example:

skey (x) * flash (x)
meaning "any flashing S key"

can be generalised to

key (x) * flash (x)
meaning "any flashing key."

A second method of generalisation is to replace constants with variables. For example:

lit (S1) * skey (S1) meaning "a lit S1 key"

can be generalised to

lit (x) * skey (x) meaning "any lit S key."

There are three types of notepad forms relating to task, key effect and indicator meaning.

When the student decides that he has successfully completed a task, he can commit this to a permanent record using the notepad. A task is defined and named by the student. Although there are some obvious tasks, such as connecting an external caller to an extension, it is left up to the student to decide when a sequence of key presses has accomplished something worth recording. An example of a task form is given below. The headings are already in the form when it appears, but the student is responsible for filling in the details.

A task can be described at two levels, either in terms of the indicator states or of the calls on the exchange. The student can request a task form for either, so when the form appears the corresponding language window is provided.

The student is also able to describe a specific task at a more general level.

Once the student has signified completion of the form, the co-learner program can examine the content and decide what comment to make. The method of examination depends on whether the student has completed an instance or a generalisation. For example, if a task instance is suggested:

- If there is no generalisation of the task on record, the instance is accepted.
- If there is a generalisation of the task and the instance is consistent with it, the instance is accepted [the generalised form could be pointed out].

- If there is a generalisation of the task but the instance is not consistent with it, the co-learner rejects the instance, pointing out where the inconsistency lies. The student can then amend the instance and re-submit it, abandon the attempt or compel the machine to accept it, overriding its objections.

The entries in the notepad are accessible at any time, so that the student can go back and update or amend an entry, either an instance or a generalisation. The co-learner program will treat the amended entry as if it were new and carry out the appropriate consistency checks outlined above.

Note that the co-learner can cooperate by itself suggesting generalisations (which it can check against the interaction history records).

Similar forms are available for describing key effects and the significance of indicator states.

A comment window is available and can be used by the co-learner to make its own suggestions (e.g., a generalisation of key effect instances or of task instances) in the comment window. At present, suggestions are only made if the student requests the co-learner to do so.

How the Co-learner Helps the Learner

The co-learner system then, as its stands at present, offers two main tools to assist the learner: the key press record window and the notepad window.

For the learner, the key press record window is intended to act as a memory aid when the learner decides a useful task goal has been accomplished. In the experimental protocols, the key press record may have been of assistance to the subjects in the following extract:

F. Now do we press on this.
    [press "Ringing External Numbers"]
    Yes we do. It's the same procedure we followed with internal.
J. Go on then, press it.
    [3 second pause]
F. And we should be able to go away...
J. Why don't we have a list of options, like we did before.
F. I don't know.
J. Why is this lit up? Try pressing that.
F. That doesn't work. Shall we try again?

In this situation, there is some doubt about exactly what has already been done. Browsing the key press record might have alleviated the problem.

The other main tool is the notepad window. This allows the learner to enter observations and generalisations of these observations. The job of the co-learner is to check that generalisations are consistent with the observations that they are intended to cover and that any new observations recorded are consistent with existing generalisations.

The following extract is an example from the protocols of one of the subjects looking to make a generalisation about two instances of connecting an external caller to an extension, but finding that the effects on the indicators are not consistent, making a satisfactory generalisation difficult to find. This is the sort of consistency check that the co-learner would be able to supply:

P. Yes, but the point was, previously, what we've got now is L2 half-black and L1 all-black. When you finished previously, L1 was half-black. It's not consistent. It's done something different in my interpretation. Unless you did something different with the mouse there's something I don't understand there, and I don't understand what those lights are telling us.

The co-learner also has a limited facility to offer its own generalisations in the comment window. Although there are no direct examples in the protocols of one subject directly asking the other for a generalisation, there are situations in which this ability would be important, such as when the two subjects are discussing hypotheses about the lighting of the L keys in the following extract:

P. Try it again and see what happens. Yes, that was there before.
<press L2, press L2, press L3, press L2, press L1>
S. That does nothing.
P. When you hit the all-black one, it toggles between... Oh.
S. How about going to that one that says 'CLG Exchange'?
P. Which ever one you press becomes the half-black, half-white.
S. What will happen then when I press this? It'll go half and make that one black.
P. It'll look like L1 and L2 have swapped. So we seem to be filling up these loop keys.
S. Let's try filling up L4.
P. We'll make L1 all-black and L4 half-black.
<press L4>
Yes, I can perceive a pattern.
S. Now what happens when we press a black one. It'll change that one.
P. Which ever one you press will swap with L4.
S. What will happen when we press another number?

Conclusions

This research into the design of a co-learner for procedural tasks has distinguished three levels of understanding:

(1) The interaction with the Merlin system where key presses can bring about particular system state changes
(2) The functional component level, which is concerned with understanding the purpose of a component within the system and the goals that can be achieved using it
(3) The task-situation level which is concerned with constructing method-plans to achieve the larger task goals for which Merlin was designed.

These distinctions imply that communication between the user and the co-learner must be possible at these levels and the notepad technique was developed for this purpose. Although the co-learner can keep a key press state change record, the notepad system allows the user to browse this record and make interpretations by segmenting and generalising so that the terminology of tasks and functions is introduced into the co-learner's vocabulary; i.e., it can start to build a task and a function space.

The co-learner has two direct ways it can assist the users, by critiquing their suggestions and by developing its own. The former technique is to check the consistency of the suggestions against the task record and inferences from it; e.g., that instances do not conflict with a given generalisation. The co-learner can contribute with some elementary planning; i.e., using identified portions from the task record (with pre- and post-conditions) in combination to "suggest" larger tasks that can be achieved.

The co-learner's main technique, though, is to consider exploratory learning as a process of generalisation/differentiation. An instance, e.g., a state change in reaction to key presses, can be generalised by replacing values with variables and/or generalising the type of key press (e.g., to a class of keys) or effect on an indicator (e.g., to a class of indicators). These generalisations can then be matched against the record, or held in store for future application, and if the inference fails for a set of instances because of over-generalisation, the conditions can be 'tuned' or its span restricted so that the rule is kept consistent with past experience shown on the work record.

The experimental data give some assurance that the co-learner technique could be a useful aid in developing users' understanding, able to extend and adapt to their interpretive viewpoints.

Acknowledgements: The research was part of a collaboration with the Computer Science department at Lancaster University and the author is grateful for the advice of Dr. John Self.

The research was supported by SERC under the ALVEY research initiative.

References

Alderman, D.L. (1978). Evaluation of the TICCIT CAI system in the Community College (Final Report). Princeton, NJ: Education Testing Service.

Bork, A.M. and Robson, J. (1972). A computer simulation for the study of waves. *American Journal of Physics* 40:1288-1294

Merrill, D.M. (1980). Learner control in computer-based learning. *Computer and Education* 4(2):77-96

Hypertext Learning Environments and Cognitive Flexibility: Characteristics Promoting the Transfer of Complex Knowledge

Michael J. Jacobson and Rand J. Spiro
University of Illinois at Urbana-Champaign
Center for the Study of Reading
174 Children's Research Center, Champaign, IL 61020

Abstract: Recent applications of educationally oriented hypertext systems have attempted to structure knowledge-rich *learning environments*, often in conceptually demanding domains and for students who are at an advanced stage of learning. A study is reported that investigated the use of recent cognitive instructional theory to structure a hypertext system to provide instruction in a complex and ill-structured content area. The results of the study revealed that the more hypertext-like treatment promoted superior *transfer* to new situations, while a hybrid hypertext/computer-based drill treatment led to higher performance on the measures of *memory* for factual knowledge. The experimental findings are discussed from the perspective of the cognitive theoretical framework that underlaid the experimental hypertext system. Instructional prescriptions for hypertext learning systems, such as the use of multiple representations of knowledge and the demonstration of critical interrelationships between abstract and case-centered knowledge components in multiple contexts, were found in the study to help promote the mastery of complex knowledge and the ability to apply this knowledge to new situations. Some implications of these findings for the development of hypertext learning environments is also discussed.

While early applications of hypertext technology often focused on providing flexible access to a wide variety of information types, newer educationally oriented hypertexts have attempted to structure knowledge-rich *learning environments* (e.g., Beeman et al., 1987; Beeman et al., 1988; Conklin, 1987; Crane and Mylonas, 1988; Dede, 1987, 1988; Jonassen, 1986, 1988; Spiro and Jehng, 1990). An important characteristic of these recently developed hypertext systems is that, unlike traditional computer-based instructional approaches which are commonly developed in instructional contexts that either are at an introductory stage of learning or are in more conceptually well-structured content areas, these hypertext learning environments are frequently developed for more advanced learning situations involving complex and cognitively challenging knowledge.

Although hypertext systems have been used recently to promote the achievement of more advanced learning goals, there has been little discussion in the hypertext literature concerning recently identified learning problems associated with more demanding instructional contexts (Jacobson and Spiro, 1991a; Spiro, Feltovich, Jacobson, and Coulson, in press; Spiro and Jehng, 1990). Serious learning failures have been identified in recent cognitive instructional research, such as the development of conceptual misunderstandings (e.g., Feltovich, Spiro, and Coulson, 1989; McCloskey, 1983; Perkins and Simmons, 1989; Spiro, Coulson, Feltovich, and Anderson, 1988) and the inability of students to transfer knowledge to relevant new situations (e.g., Bransford, Franks, Vye, and Sherwood, 1989; Brown, 1989; Gick and Holyoak, 1987; Schoenfeld, 1983). Unfortunately, many instructional applications of hypertext have been developed in an a theoretical manner and have often not considered learning problems such as these that confront learners of complex knowledge (Jacobson and Spiro, 1991a; Spiro et al., in press; Spiro and Jehng, 1990).

The major goal of this paper is to discuss features of a theory-based hypertext learning environment employed in a study that investigated knowledge acquisition and transfer issues in a complex and ill-structured domain. The results of this study revealed significant knowledge transfer effects associated with a new approach to structuring an instructional hypertext system. As a detailed account of the research methodology and findings may be found in Jacobson and Spiro (1991b), this paper will focus on the theoretical framework used to structure the hypertext system that contributed to the attainment of knowledge transfer.

Theoretical Framework and Research Issues

Cognitive Flexibility Theory

Cognitive Flexibility Theory (CFT) (Spiro et al., 1988) is a theory of learning and instruction that seeks to provide a framework for both understanding and promoting the advanced acquisition of knowledge, particularly in complex domains that possess a high degree of conceptual irregularity or ill-structuredness. (Due to space limitations, this section will provide only a brief discussion of CFT. For a fuller explication of this theory, see Spiro, Vispoel, Schmitz, Samarapungavan, and Boerger, 1987; Spiro et al., 1988; and Spiro and Jehng, 1990.) An important aspect of the theory, that is a central research issue in this study, is its concern with *knowledge transfer*.

An analysis of learning failures in several recent studies has identified various forms of *oversimplification of complex knowledge* as a significant contributing factor (Coulson, Feltovich, and Spiro, 1990; Feltovich et al., 1989; Myers, Feltovich, Coulson, Adami, and Spiro, 1990; Spiro, Feltovich, Coulson, and Anderson, 1989). These sources of failure in understanding are also believed to contribute to deficiencies in transfer ability. To address these problems, CFT makes a number of instructional prescriptions, such as: use of *multiple knowledge presentations* (e.g., multiple themes, analogies, case examples, lines of argument); explicitly *linking and tailoring concepts to practice or case examples* (i.e., situating conceptual knowledge in contexts that are similar to those required for the application of the knowledge); *incrementally introducing complexity* in small, cognitively manageable units; stressing the *interrelated and web-like nature of knowledge* (instead of isolated and compartmentalized knowledge); and encouraging the *assembly* of appropriate knowledge from various conceptual and case sources (rather than the intact retrieval of previously memorized information) (Spiro et al., 1988; Spiro and Jehng, 1990). The use of instructional features such as these are predicted by CFT to help the learner cultivate a multifaceted and flexible cognitive representation of complex knowledge. Such cognitive representations should enhance knowledge transfer by increasing the likelihood that an appropriate compilation of knowledge can be constructed to fit the understanding or problem-solving needs for the wide variety of case characteristics associated with ill-structured domains.

Research Issues

There is preliminary experimental evidence from two noncomputer studies that procedures based on Cognitive Flexibility Theory in fact promote the ability of students to acquire and transfer complex knowledge (Hartman and Spiro, 1989; Spiro et al., 1987). In addition, CFT design prescriptions have also been used to develop several prototype instructional hypertext systems (Spiro et al., 1988; Spiro and Jehng, 1990). The present study was intended to extend this research and development program to specifically test selected CFT theoretical and design issues associated with the use of hypertext learning environments.

The main experimental hypothesis of this study was that a hypertext approach incorporating several design features based on CFT would promote the acquisition of complex material (the social impact of technology) in a manner that would enhance the learners' ability to apply the knowledge in cognitively demanding and novel knowledge application tasks. A control condition—a hybrid hypertext/computer-based drill—was developed which employed a number of antithetical design features that presented the same instructed knowledge in a more rigid and decontextualized manner. While the latter treatment was predicted to foster the efficient and effective acquisition of factual knowledge in the domain, it was expected that subjects in this group would perform at a lower level on tasks requiring knowledge transfer (i.e., the application of instructed knowledge to conditions unlike those found in the initial learning) than would the subjects receiving the experimental hypertext treatment.

Overview of "Social Impact of Technology" Hypertext Study

Experimental Features

The social impact of technology was selected as the topic for the instructional content. This is a complex domain that is conceptually ill-structured due to the multidisciplinary focus of the literature and the rich and dynamic conditions that characterize real world applications of this knowledge. Thirty four subjects were randomly assigned to the two main treatment groups for a study that was administered in two hour sessions over a period of four days. An initial period of knowledge acquisition, referred to as *Reading Time*, provided an introductory exposure to the subject matter using similar hypertext materials read by both groups. A period of more advanced study was provided during *Study Time* in which the contrasting treatments for the two groups were administered. The main component of the treatment for the experimental group, referred to as *Thematic Criss-Crossing*, involved a hypertext environment in which abstract and case-centered components of the domain were studied from a variety of intellectual perspectives that highlighted different aspects of the knowledge. The control condition consisted of a computer-based drill that provided practice on factual items drawn from *Reading Time* and on items requiring the recognition of abstract knowledge components in previously read passages. The primary evaluation tasks, which were administered on the second and the fourth sessions, consisted of short answer tests on the acquisition content and written essays requiring the subjects to analyze and resolve issues posed in new social impact of technology situations.

Major Results

The short answer items were intended to measure the acquisition of factual knowledge associated with the instructional content. As predicted, the hybrid hypertext/computer-based drill group achieved significantly higher scores on this test of lower-level domain knowledge (see Table 1). We feel that there were several aspects of the computer-based drill treatment that contributed to this finding, such as the *simplification* and *abstraction* of the domain knowledge resulting from studying the material using specific multiple-choice questions on factual and theme content and the implicit emphasis on *rote* memory of the material resulting from recycling missed items for further practice in the drill format (see also Jacobson and Spiro, 1991a).

Table 1

Mean Scores and Standard Deviation for Short Answer and Problem-Solving Essay

	Session 2		Session 4	
Treatment Group	SA	ESSAY	SA	ESSAY
Drill[a]				
Mean	52.45	7.65	52.72	7.24
SD	10.01	2.87	10.38	2.56
Thematic Criss-Crossing[a]				
Mean	47.55	8.41	47.23	9.47
SD	9.66	3.54	9.09	2.35

Note: SA - Short Answer. Short Answer results reported as Normalized T-scale scores (Mean = 50, SD = 10). Possible Essay scores range from 0 to 14. [a]n = 17.

While there was no difference between the groups on the judges' scores for the more cognitively demanding Problem-Solving Essay task administered at the end of Session 2, the Thematic Criss-Crossing group achieved significantly higher essay scores by Session 4. The delayed effectiveness of the Thematic Criss-Crossing treatment was expected as it was thought that a critical period of time would be required by the subjects to assimilate the multiple and interrelated perspectives of the knowledge presented by this treatment. Subjects receiving the Thematic Criss-Crossing treatment thus were better able to apply their acquired knowledge of the social impact of technology in an unfamiliar within-domain problem-oriented situation.

Hypertext and Learning Complex Knowledge for Transfer: Theoretical and Design Considerations

A short duration study such as this (over a two week period with about four hours of instruction) is but a snap-shot of advanced learning in a complex and ill-structured knowledge domain. It is therefore notable that significant experimental effects were achieved in the limited amount of time available for the study that were highly consistent with the study's cognitive theoretical framework. This section will examine the experimental findings from the perspective of five Cognitive Flexibility Theory instructional prescriptions (discussed earlier) that were incorporated into the hypertext system used in this study. The experimental Thematic Criss-Crossing group received all five of these prescriptions while the hybrid hypertext/computer-based drill received only three of the prescriptions, along with a computer-based drill.

Cognitive Flexibility Hypertext Features Common to Both Experimental and Control Groups

Use multiple representations of knowledge. A central recommendation of CFT concerns the use of multiple knowledge representations to help avoid the oversimplification of complex knowledge. The hypertext materials used during Reading Time were intended to represent the complexity of the social impact of technology domain through the use of multiple case examples and multiple thematic or conceptual perspectives. Six cases were written as the primary hypertext knowledge nodes that described different examples of the social impact of technology, such as a case describing an attempt to develop an electric car in France during the 1970s and a case discussing the use of human subjects for medical research on the development of an artificial heart. Multiple conceptual perspectives were provided through the use of six broad-scope and partially overlapping thematic dimensions of analysis that were identified from the literature, such as *Freedom - Control* and *Community - Alienation*.

Link concepts to case examples. The six cases were subdivided into *minicases* that served as the primary instructional units/hypertext nodes. Each minicase consisted of a short section of text describing a portion of the case, a list of relevant themes, and theme commentaries (see Spiro and Jehng, 1990). The text of each minicase was analyzed to determine which of the six social impact of technology themes were relevant; applicable themes were placed in a Theme List node for the subject to read. Theme commentaries were written that were intended to be representative of expert knowledge in this domain. The commentaries were specific to every minicase for each of the listed themes. The theme commentaries explicitly linked the generic definitions of the thematic perspectives (i.e., abstract knowledge components) to the specific features of text sections used in the minicases (i.e., case-centered knowledge components).

Incrementally introduce complexity. The minicase sections represented small and cognitively manageable units of study that still reflected aspects of the domain's conceptual complexity and ill-structuredness even at the beginning of the instructional treatment. The instructional sequence provided increasing amounts of complexity as the subject read more social impact of technology cases. Also, because of the different combination of themes associated with each minicase, variability of conceptual or thematic coverage could be seen both within and between cases. Concepts were thus *not* presented in an oversimplified or abstracted manner initially as in many curricular approaches, with the potential problem that the early simplifications would interfere with learning domain complexities at a later phase of instruction (Feltovich et al., 1989; Spiro et al., 1987; Spiro et al., 1988, 1989). Instead, the early instructional sequence reflected a moderate degree of the richness associated with the domain that was then elaborated on with later Study Time learning activities.

Cognitive Flexibility Hypertext Features of the Experimental Thematic Criss-Crossing Group

This section discusses two additional tenets of Cognitive Flexibility Theory, the *web-like nature of knowledge* and *knowledge assembly from different conceptual and case sources*, that were implemented in the Thematic Criss-Crossing treatment administered to the experimental group during Study Time. It is important to note that while these two additional prescriptions were added during Study Time, Thematic Criss-Crossing continued to utilize (in slightly different ways) the three prescriptions discussed earlier. The *multiple representations of knowledge* and *incrementally introducing complexity* prescriptions were instantiated through the use of multiple combinations of themes from which to explore the minicase knowledge base in novel and intricate ways (i.e., thematic criss-crossing), and the *linking of concepts to case examples* was continued in a similar manner as in Reading Time through the use of the same theme lists and thematic commentaries (although subjects were required to read only selected theme commentaries associated with a particular Thematic Criss-Crossing combination). The next two sections briefly discuss the additional CFT prescriptions employed in the Thematic Criss-Crossing treatment.

Stress interrelated and web-like nature of knowledge. As noted earlier, Thematic Criss-Crossing required the subjects to reread minicases that exemplified different combinations of themes. It was hoped that this procedure would help demonstrate case-based conceptual interrelationships in the domain knowledge that may have been unnoticed when initially studying the material in Reading Time. For example, reading minicase A with the themes "x y z," minicase B with themes "u x y," and minicase C with themes "v z" may not suggest any underlying knowledge interrelationships to a novice in the domain. However, considering themes "x y" together demonstrates knowledge linkages between minicases A and B, while a consideration of theme "z" illustrates an overlap between minicases A and C. Each combination of themes allows a different traversal of the minicases that comprise the knowledge base of the hypertext. Such a theme-based exploration of the minicases is intended to highlight many different facets of the domain knowledge. In all, the Thematic Criss-Crossing subjects read 15 different thematic combinations over the four experimental sessions to implement the CFT prescription to demonstrate the web-like structure of the conceptual and case interrelations in the instructional domain.

Encourage assembly of knowledge from different conceptual and case sources. The Thematic Criss-Crossing treatment was designed to provide an externalized, technology-based procedure that explicitly demonstrated how knowledge could be assembled from different conceptual and case sources, a procedure that could be considered analogous to knowledge representation schemes observed in experts. For example, protocols of experts engaged in problem-solving tasks in a variety of domains have documented their ability to flexibly apply knowledge and solve problems in a domain (e.g., Chi, Glaser, and Farr, 1988). It may be that the *assembly* of knowledge from various conceptual and case sources rather than just the intact retrieval from memory of precompiled knowledge structures (Spiro et al., 1988) is a component underlying the flexible knowledge application and problem solving performance observed in experts. Part of the effectiveness of the Thematic Criss-Crossing treatment thus may be due to the externalized *modeling of knowledge assembly* demonstrated through the computer re-editings of the link relationships between the minicases based on different theme combinations. With this procedure, the student directly experiences the combination and recombination of knowledge components that, internalized over time, undergirds the development of cognitive flexibility and the improved ability to transfer knowledge to new situations.

Implications for the Design of Hypertext Learning Environments

There are important implications for the design and use of instructional hypertext systems that are suggested by the findings of significant transfer effects associated with the experimental Thematic Criss-Crossing treatment on the Problem-Solving Essay task. While a full consideration of these issues is beyond the scope of this paper, a critical implication of the study is that *merely providing hypertext links to different information nodes is not sufficient to promote the transfer of knowledge to new situations*. Both groups had an equal opportunity to learn abstract and case-centered components of the domain knowledge with the hypertext materials used during Reading Time and both groups also received further study in the domain during the contrasting Study Time treatments. As noted earlier, the superior performance of the Thematic Criss-Crossing group on the Problem-Solving Essay suggests that the ability to acquire and transfer complex domain knowledge requires *special design parameters* for the hypertext learning environment, such as presenting the knowledge in multiple and interconnected ways. Future research is obviously needed to study the effects of such theory-based design parameters on the effectiveness of hypertext learning environments in a variety of content areas and instructional situations.

Conclusion

This paper discussed a number of theory-based features of an approach to structuring a hypertext system that was used in a recent study of learning and knowledge transfer in a complex and ill-structured content area. Significant transfer effects were found for the experimental hypertext treatment that employed recent cognitive instructional theory prescriptions such as the use of *multiple knowledge representations*, depicting the *interconnected and web-like* structure of abstract and case-centered knowledge components, and stressing the *assembly* of knowledge from different conceptual and case sources. The rigorous evaluation of instructional hypertext design principles is still a relatively new area of cognitive research. Given the proper attention to critical theoretical and empirical issues associated with learning complex knowledge, hypertext learning environments show promise of being a valuable instructional supplement to other types of cognitively oriented computer-based learning systems such as microworlds, intelligent tutoring systems, and empowering environments.

This research was supported in part by grants from the University of Illinois Graduate College and College of Education to the first author and in part from funding by the Office of Educational Research and Improvement (OEG 0087-C1001) and the Army Research Institute Office of Basic Research (MDA903-86-K-0443) to the second author. The opinions expressed in this paper are not necessarily those of the funding agencies.

Acknowledgements: This paper is based on portions of the first author's dissertation research. The first author gratefully acknowledges help and support he received from research programmer Susan Ravlin and from the members of his dissertation committee: Rand J. Spiro (Thesis Director), Michael L. Waugh, James A. Levin, J. R. Dennis, and Klaus Witz. Both authors extend thanks to James A. Levin who made helpful comments on an earlier version of this paper.

References

Beeman, W. O., Anderson, K. T., Bader, G., Larkin, J., McClard, A. P., McQuillan, P. J., and Shields, M. (1987). Hypertext and pluralism: From lineal to non-lineal thinking. In *Hypertext '87 Papers* (pp. 67-88). Chapel Hill: The University of North Carolina

Beeman, W. O., Anderson, K., Bader, G., Larkin, J., McClard, A. P., McQuillan, P., and Shields, M. (1988). *Intermedia: A case study of innovation in higher education* (Final report to the Annenberg/CPB Project). Providence, RI: Brown University, Office of Program Analysis, Institute for Research in Information and Scholarship.

Bransford, J. D., Franks, J. J., Vye, N. J., and Sherwood, R. D. (1989). New approaches to instruction: Because wisdom can't be told. In S. Vosniadou, and A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 470-497). Cambridge: Cambridge University Press.

Brown, A. L. (1989). Analogical learning and transfer: What develops? In S. Vosniadou, and A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 369-412). Cambridge: Cambridge University Press.

Chi, M. T. H., Glaser, R., and Farr, M. J. (Eds.). (1988). *The nature of expertise*. Hillsdale, NJ: Lawrence Erlbaum.

Conklin, J. (1987). Hypertext: An introduction and survey. *IEEE Computer*, 20(9), 17-41.

Coulson, R. L., Feltovich, P. J., and Spiro, R. J. (1989). Foundations of a misunderstanding of the ultrastructural basis of myocardial failure: A reciprocation network of oversimplifications. *The Journal of Medicine and Philosophy*, 14, 109-146.

Crane, G., and Mylonas, E. (1988). The Perseus project: Interactive curriculum on classical Greek civilization. *Educational Technology*, 28(11), 25-32.

Dede, C. J. (1987). Empowering environments, hypermedia and microworlds. *The Computing Teacher*, 15(3), 20-24, 61.

Dede, C. J. (1988). The probable evolution of artificial intelligence based educational devices. *Technological Forecasting and Social Change*, 34, 115-133.

Feltovich, P. J., Spiro, R. J., and Coulson, R. L. (1989). The nature of conceptual understanding in biomedicine: The deep structure of complex ideas and the development of misconceptions. In D. Evans, and V. Pate (Eds.), *The cognitive sciences in medicine* (pp. 113-172). Cambridge, MA: MIT Press (Bradford Books).

Gick, M. L., and Holyoak, K. J. (1987). The cognitive basis of knowledge transfer. In S. M. Cormier, and J. D. Hagman (Eds.), *Transfer of learning: Contemporary research and applications* (pp. 9-46). New York : Academic Press.

Hartman, D. K., and Spiro, R. J. (1989). *Explicit text structure instruction for advanced knowledge acquisition in complex domains: A post-structuralist perspective*. Paper presented at the annual meeting of the American Educational Research Association, San Francisco, CA.

Jacobson, M. J., and Spiro, R. J. (1991a). *A framework for the contextual analysis of computer-based learning environments*. Manuscript submitted for publication.

Jacobson, M. J., and Spiro, R. J. (1991b). *Hypertext learning environments, cognitive flexibility, and the transfer of complex knowledge: An empirical investigation*. Manuscript submitted for publication.

Jonassen, D. H. (1986). Hypertext principles for text and courseware design. *Educational Psychologist*, 21(4), 269-292.

Jonassen, D. H. (1988). Designing structured hypertext and structuring access to hypertext. *Educational Technology*, 28(11), 13-16.

McCloskey, M. (1983). Naive theories of motion. In D. Gentner and A. L. Stevens (Eds.), *Mental models*. Hillsdale, NJ: Lawrence Erlbaum.

Myers, A. C., Feltovich, P. J., Coulson, R. L., Adami, J. F., and Spiro, R. J. (1990). Reductive biases in the reasoning of medical students: An investigation in the domain of acid-base balance. In B. Bender, R. J. Hiemstra, A. J. J. A. Scherbier, and R. P. Zwierstra (Eds.), *Teaching and assessing clinical competence*. Groningen, the Netherlands: BoekWerk Publications.

Perkins, D. N., and Simmons, R. (1989). Patterns of misunderstanding: An integrative model for science, math, and programming. *Review of Educational Research*, 58, 303-326.

Schoenfeld, A. H. (1983). Beyond the purely cognitive: Belief systems, social cognitions, and metacognitions as driving forces in intellectual performance. *Cognitive Science*, 7, 329-363.

Spiro, R. J., Coulson, R. L., Feltovich, P. J., and Anderson, D. K. (1988). Cognitive flexibility theory: Advanced knowledge acquisition in ill-structured domains. *Tenth Annual Conference of the Cognitive Science Society* (pp. 375-383). Hillsdale, NJ: Lawrence Erlbaum.

Spiro, R. J., Feltovich, P. J., Coulson, R. L., and Anderson, D. K. (1989). Multiple analogies for complex concepts: Antidotes for analogy-induced misconception in advanced knowledge acquisition. In S. Vosniadou, and A. Ortony (Eds.), *Similarity and analogical reasoning* (pp. 498-531). Cambridge: Cambridge University Press.

Spiro, R. J., Feltovich, P. J., Jacobson, M. J., and Coulson, R. L. (in press). Cognitive flexibility, constructivism, and hypertext: Random access instruction for advanced knowledge acquisition in ill-structured domains. *Educational Technology*.

Spiro, R. J., and Jehng, J. C. (1990). Cognitive flexibility, random access instruction, and hypertext: Theory and technology for the nonlinear and multidimensional traversal of complex subject matter. In D. Nix, and R. J. Spiro (Eds.), *The "Handy" project: New directions in multimedia instruction* (pp. 163-205). Hillsdale, NJ: Lawrence Erlbaum.

Spiro, R. J., Vispoel, W. L., Schmitz, J. G., Samarapungavan, A., and Boerger, A. E. (1987). Knowledge acquisition for application: Cognitive flexibility and transfer in complex content domains. In B. K. Britton, and S. M. Glynn (Eds.), *Executive control processes in reading* (pp. 177-199). Hillsdale, NJ: Lawrence Erlbaum.

Using Button Theory to Structure Student-Teacher Interaction in Computer-Based Learning Environments

Menachem Jona, Benjamin Bell, and Lawrence Birnbaum
The Institute for the Learning Sciences
Northwestern University, 1890 Maple Avenue, Evanston, Illinois 60201

Abstract: This paper presents *Button Theory*, developed to allow students using computer-based learning environments to have as much control as possible over what they see, hear, and learn. The two principal goals of the theory are, first, to provide a taxonomy of the ways that students might usefully interact with and control a computer-based teacher, and second, to provide a natural mechanism by which they may exercise that control. We have developed a small but comprehensive set of messages that students would find it useful to convey to a teacher during a tutorial interaction. As the name of the theory indicates, rather than using natural language, the mechanism we provide to enable students to express these messages is a set of buttons presented iconically on the computer screen, each button corresponding to one message.

Summary of Button Theory

One of the more serious problems facing education today is that students lack control over the instruction they receive (see, e.g., Schank & Jona, 1991). When students feel that their learning needs are not being attended to, the result is boredom, frustration, and a dislike for learning in general. This is especially true in computer-based learning environments, where technological limitations, aggravated in some cases by poor design decisions, have historically put the student in an extremely passive role (see, e.g., Papert, 1980). This paper introduces Button Theory, developed in order to allow students using computer-based learning environments to have as much control as possible over what they see, hear, and learn.

Motivation for Button Theory

The two principle goals of Button Theory are, first, to provide a taxonomy of the ways that students might usefully interact with and control a computer-based teacher, and second, to provide a natural mechanism by which they may exercise that control. We have developed a small but comprehensive set of messages that students would find it useful to convey to a teacher during a tutorial interaction. These messages are organized into three groups, **feelings**, **questions**, and **control**, reflecting the three natural domains of discourse in a computer-based learning environment: The student may wish to talk about himself (**feelings**), about the domain or task of instruction (**questions**), or about the teaching itself (**control**). The **feelings** group contains messages about how the student is feeling about the lesson, allowing expressions of boredom, excitement, or disbelief. The **questions** group, as its name implies, contains various questions a student is likely to ask (e.g., Why?, What's the point?, etc.). The third group, **control**, allows the student to change the progression of the lesson to suit his or her needs, speeding up when the material is too easy, backing up when it is too difficult, or getting more detail about something interesting.

As the name of the theory indicates, rather than using natural language, the mechanism we provide to enable students to express these messages is a set of buttons presented iconically on the computer screen, each button corresponding to one message. The user can "press" any of these buttons using a mouse-pointer. This "Button Pad" can be thought of as analogous to the remote-control clicker of a TV or VCR, except that instead of controlling the volume or the channel, it allows the student to manage the instruction he or she is receiving.[1]

In addition to its goal of giving students more control over the instruction they receive, Button Theory also reflects our desire to sidestep, as much as possible, the need for student modelling in computer-based learning environments. The difficulty of constructing meaningful student models has become nearly a truism among researchers engaged in the construction of computer-based learning environments. Button Theory attempts to minimize the amount of student modelling required by encouraging students to express their internal state through the Button Pad. Providing an easy-to-use mechanism for students to communicate with the teacher will allow a commensurate reduction in the amount of educated guessing the teacher will have to do about what's going on with the student. Needless to say, the need for student models will not be completely eliminated. We discuss an example of the kind of modelling required to interpret the meaning of button presses in Section 2 below.

Taxonomy of Student Messages

The first step in developing a theory of communicative interaction in tutorial situations must be to develop a taxonomy of kinds of communicative acts that a student might be expected to perform in such situations, so as to characterize the class of expected inputs. Moreover, this characterization must be functional in nature, that is, it must facilitate a determination of the intended meaning of a communicative act, and of an appropriate response. We must, in other words, develop a representation theory that can be used to categorize student inputs in terms of both content and intent. This is true regardless of whether the ultimate application of the theory will be in the development of the sort of iconic interface we are pursuing, or in natural language processing.

As described above, the types of communications that might take place between a student and a computer teacher can be divided into the three categories of **feelings**, **questions**, and **control**. These categories are determined by the three naturally-occurring domains of discourse that exist in any interaction between a student and a computer-based teacher, namely (a) the student, (b) the teacher/program, and (c) the domain and/or task being taught. Within each of the above categories, we have selected what we believe to be the five most important, most frequently used communicative acts, and assigned each to an iconic button on the Button Pad (see Figure 1).

Case Study

To investigate the usefulness of Button Theory we have incorporated the Button Pad interface in several prototype computer-based learning environments built at Northwestern

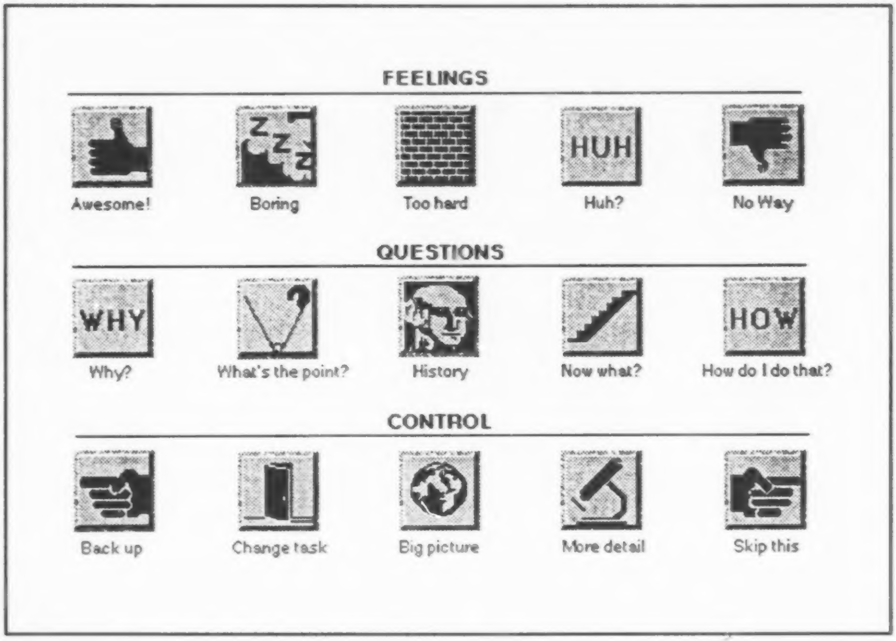

FEELINGS
Awesome!
Boring
Too hard
HUH
Huh?
No Way
QUESTIONS
WHY
Why?
What's the point?
History
Now what?
HOW
How do I do that?
CONTROL
Back up
Change task
Big picture
More detail
Skip this

Figure 1. Button icons from the Button Pad as they might appear on the screen.

University's Institute for the Learning Sciences, most of which use only a subset of the fifteen buttons described above. In this section, we will provide a detailed case study of the use of as many buttons as possible, based on one such program, called *ChimpWorld*. The ChimpWorld system is built on top of a detailed simulation of chimpanzee societies and is designed to teach the fundamentals of primate social behavior by actually allowing the student to play the role of a chimpanzee in the simulation.[2] We envision the target audience of this system to be students in a junior high or introductory high school biology class.

In this section we first provide an annotated transcript of a sample interaction with the ChimpWorld system. Next, we describe the various contextual variables which the system uses in selecting a response to a given button press. Finally, we present some of the teaching strategies that the system uses in responding to button presses.

Sample Run

The transcript below comes from an actual interaction with the ChimpWorld system and shows what a first-time user might encounter. The student's actions and button presses are shown in **boldface**. Bracketed phrases describe actions taken by the student in the simulation. Phrases in italics provide additional information about other actions occurring in the simulation.

Welcome to ChimpWorld! ChimpWorld will help you learn about the social behavior of chimpanzees by letting you actually play the role of a chimpanzee. You will use the Action Palette at the upper right corner of the screen to control your chimp. If you need help press one of the buttons on the Button Pad. Let's start now. In the ChimpWorld window above, you are the chimp on the right side of the window. Go for it!

Now what?

What are the other chimps doing?

Huh?
[Click on lettuce]

That is some lettuce, a kind of food that chimps eat.

How do I do that?

The tutor infers that the student is referring to the action of eating because the student's last question was about food and the previous response mentions eating explicitly.

Before you can eat you have to find out if you are sufficiently hungry and be near some food.

How do I do that?

In order to find out if you are sufficiently hungry you have to look at the Hunger graph in your Motivations Window.

[LOOK-AT-HUNGER-GRAPH]
Now what?

Right. The last thing you need to do before you can eat is to be near some food.

How do I do that?

In order to be near some food you can be right next to some food or you can walk over to some food.

Now what?

You need to walk over to some food.

History.

So far you've done one of the things you need to do before you can eat. You have been successful in finding out if you are sufficiently hungry.
Now what?

Before you can walk over to some food you have to press the WALK button on the Action Palette and click the mouse on a destination.

[PRESS-WALK]
[SELECT-LOCATION-TARGET]

The student's chimp walks over to the lettuce and sits there.

Now what?

OK. Now that you've satisfied all the prerequisites in order to eat you should press the EAT button on the Action Palette.

[PRESS-EAT]
Now what?

That's right. Now click the mouse on some food.

[SELECT-FOOD-TARGET]

The dominant chimp, sitting nearby, walks over to the student's chimp, and begins to threaten it.

Huh?
[Click on chimp threatening]

That chimp is threatening you.

Since the student chimp has not responded, the other chimp begins to beat on the student chimp.

Why?

Story: Dominance hierarchies

Chimpanzee societies have what is known as a dominance hierarchy. A dominance hierarchy is like a pecking-order: it establishes the rank of each chimp within the society. The most dominant chimp in the hierarchy, almost always a male, is known as the Alpha Male. Dominant males try to control which chimps get food and which chimps are allowed to mate.

Would you like to see a video about the hierarchy in chimpanzee societies? If so, press the Play button on the video clicker.

Student presses Play, watches video that introduces ideas of a chimp dominance hierarchy.

More detail.

Another story is selected that relates more closely to what just happened to the student.

Story: Control of food

One way that dominant chimps exert their power is by controlling access to food by less dominant chimps. They do this for several reasons. First, by controlling the food they increase the chances of survival for themselves and their offspring. Second, food can be used as a political tool. Chimps that ally themselves with the dominant chimp can gain access to food, while others can not.

Press the Play button on the video clicker if you would like to see a video about how dominant chimps control food.

Student presses Play and watches a video that focuses on the control of food by dominant males. The video presents a situation similar to what just occurred in the simulation: a less dominant chimp getting beaten up and chased away from food.

[interaction continues...]

Situational Predicates

The ChimpWorld tutor keeps track of several contextual variables which it uses in selecting a response to each button press. The system uses predicates to test the value of these variables as the need arises. Undoubtedly, as the sophistication of the system increases, more contextual variables will be needed. Furthermore, there is no reason to expect that these same contextual variables will be important in every kind of teaching system. Our hope, however, is that we will be able to arrive at a "core set" of contextual variables. As other systems are built, they can add to and/or specialize this core set.

These contextual variables constitute, in part, a simple student model which the system uses in deciding on the most appropriate response to a given button press. The important thing to note here is that the design of this student model is functionally driven by the need to disambiguate button presses. The two assumptions which underlie this simple student model are (a) the student doesn't know about something until he or she is told about it, and (b) once told about something, the student does not forget or confuse it. While these assumptions are admittedly unrealistic, they do allow us to construct a simple, yet surprisingly powerful discourse generation process. For example, by checking the knowledge preconditions of a particular item to see if they have been presented to the student or not, we can easily decide whether or not to include a discussion of those preconditions in the tutorial dialog.

The contextual elements employed by Button Theory can be taxonomized into the following three groups:

- **student goals**—used to represent the various actions a student might pursue in the simulation. Examples from the ChimpWorld system are the predicates CURRENT-GOAL-JUSTIFIED and CURRENT-GOAL-NOT-JUSTIFIED. When the student is given an explanation of why he or she needs to pursue a certain subgoal, that subgoal is marked as having been "justified." Justified in this context means relating a subgoal to the goal for which it is a precondition.
- **student knowledge**—used to represent student's exposure to the various actions and objects found in the simulation. Because the student will be attempting to perform many of the same actions that he or she observes other chimpanzees performing, the representation of these objects and actions is tightly linked to the goal/sub-goal hierarchy mentioned above. For example, the ChimpWorld system includes the predicates OBJECT-NOT-DESCRIBED and OBJECT-ALREADY-DESCRIBED. When the student is given a description of what a particular object in the simulation is, that object is marked as having been described. Described in this context means answering the question "What is this thing?"
- **communication history**—This group can itself be subdivided into two subgroups:

  - **past student actions**—In selecting a response to a given button press, the ChimpWorld system refers to a history of past events. These events include, for example, past button presses by the student, and whether a button press was followed by a mouse-click specifying the referent of that button press. In ChimpWorld, the predicate REPEATED-LAST-BUTTON-PRESS tests whether the student pressed the same button as he or she did last time.
  - **past teacher actions**—The ChimpWorld system uses past actions by the system itself in selecting the most appropriate response to a button press. At this point, only the last teaching strategy triggered is used in selecting a response to the current button press. For example, the predicate LAST-STRATEGY = TELL-STORY tests whether one of the story-telling teaching strategies (*i.e.,* tell a story or show a video) was the last one triggered.

A description of the complete set of predicates appears in (Jona, Bell, & Birnbaum, 1991).

Button Responses

Using the situational predicates detailed above, the system selects an appropriate teaching strategy with which to respond. This section illustrates the strategies the tutor chooses in response to several example button presses. The complete set of responses currently used in the ChimpWorld system can be found in (Jona et al., 1991).

HOW-DO-I-DO-THAT-BUTTON

Context:	NO-CURRENT-GOAL ACTION-NOT-EXPLAINED
Interpretation:	The student doesn't know how to perform an action. The steps in doing the action have not been previously explained.
Strategy:	EXPLAIN-FIRST-STEP-AND-SET-GOAL
Description:	Explain in a general way how to begin to do an action. Set the tentative action focus as the current goal. Mark the action as having been explained.

Context:	CURRENT-GOAL
Interpretation:	The student doesn't know how to perform the next step of an action. Prior steps in doing the action have been previously explained.
Strategy:	EXPLAIN-NEXT-STEP-AND-SET-GOAL
Description:	Explain how to continue doing an action. Set the current goal to be the next step of the action. Mark the action as having been explained.

WHY-BUTTON

Context:	CURRENT-GOAL CURRENT-GOAL-NOT-JUSTIFIED
Interpretation:	The student doesn't know an action is necessary.
Strategy:	EXPLAIN-WHY-ACTION-NECESSARY
Description:	Explain why the current action is necessary by connecting it to the goal which it serves.

Context:	CURRENT-GOAL CURRENT-GOAL-JUSTIFIED
Interpretation:	The student doesn't know an action is necessary.
Strategy:	EXPLAIN-WHY-ACTION-NECESSARY-BASED-ON-ACTION
Description:	Explain why the current action is necessary by connecting it to the goal which it serves.

Context:	WHY-BUTTON LAST-STRATEGY = HANDLE-FAILURE
Interpretation:	A typical failure just occurred and the student doesn't know why it happened.
Strategy:	EXPLAIN-FAILURE
Description:	Explain why a failure occurred by telling stories if available.

Future Work and Conclusions

We are currently investigating a number of empirical questions raised by Button Theory, including such issues as the optimal number of buttons, which messages are most necessary, what kinds of responses students expect when they press a given button, what kinds of responses human teachers actually give, and how well these latter two correlate. We would also like to investigate the extent to which the buttons affect both the quantity and quality of questions asked by students, the hypothesis being that, rather than constraining the kinds of questions asked, the Button Pad will actually encourage not only more question asking, but also better question asking (Schank, 1986, 1988). We hope to report the findings of these experiments in the near future.

From a theoretical perspective, it should be apparent that the problem of correctly interpreting and responding to a given button press by a given student in a given context is, in its full extent, as difficult a problem as natural language understanding. In fact, it is the same problem. Our hope is that the framework provided by Button Theory will permit us to tackle this problem incrementally, starting with simple responses for a subset of the buttons, and gradually increasing the sophistication of the interpretive mechanisms underlying each button as necessary. The key point is that even with rather simple mechanisms, this framework enables surprisingly rich interactions. These interactions are only as successful as the taxonomy upon which they are based; but even the rather straightforward analysis of student communicative needs offered by Button Theory provides a great deal of leverage.

Notes

1. See Schank & Jona (1991) for a fuller discussion of this analogy and its implications.
2. The simulator is being built by Lucian Hughes as part of his Ph.D. research. The tutoring component is the work of the first author. The CHIMPWORLD system runs on a Macintosh IIfx with 20MB RAM and uses 3D graphic animation and on-screen video.

Acknowledgements: Button Theory was initially developed in a seminar led by Roger Schank, and including Allan Collins, Gregg Collins, Paul Cooper, Danny Edelson, Bill Ferguson, Enio Ohmaye, Andrew Ortony, and Chris Riesbeck, as well as the authors, among others. We would like to thank Lucian Hughes and Ira Hyman for many useful discussions and Ann Jacobs for designing the icons currently used in the Button Pad.

This research was supported in part by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058 and the Office of Naval Research under contract N00014-90-J-4117, by the Office of Naval Research under contract N00014-J-1987, and by the Air Force Office of Scientific Research under contract AFOSR-89-0493. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner, and from IBM.

References

Jona, M., Bell., B., & Birnbaum, L. (1991). *Button Theory: A Taxonomic Framework for Student-Teacher Interaction in Computer-Based Learning Environments*. Technical Report 12, Institute for the Learning Sciences , Northwestern University.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas.* New York: Basic Books.

Schank, R.C., & Jona, M.Y. (1991). Empowering the student: New perspectives on the design of teaching systems. *The Journal of the Learning Sciences*, *1*, 7-35.

Schank, R.C. (1988). *The Creative Attitude: Learning to ask and answer the right questions*. New York: Macmillan.

Schank, R.C. (1986). *Explanation Patters: Understanding Mechanically and Creatively*. Hillsdale, NJ: Lawrence Erlbaum.

Environments for Incidental Learning: Taking Road Trips Instead of Memorizing State Capitals

Alex Kass and David Guralnick
Institute for the Learning Sciences and
Department of Electrical Engineering and Computer Science
Northwestern University, Evanston, Illinois 60201

Abstract: Certain things are boring to study directly, even though knowing them is necessary to perform fun and interesting tasks. The best way to teach these subjects, therefore, is to provide environments in which the students can perform such tasks, and in which they can have ready, timely access to the knowledge they need. In this paper we describe two incidental learning environments that we are building: One teaches United States geography by allowing the student to take simulated car trips. The other teaches about social issues by allowing students to produce their own TV news program. We discuss several issues that must be faced when designing any incidental learning environment, including integrating tasks and teaching goals, guiding and coaching, and providing access to knowledge.

Introduction

Some things are very dull to study directly even though knowing them is necessary in order to perform some fun and interesting tasks. For instance, learning geography in school is often a boring exercise in rote memorization. However, people who travel frequently come to know geography without trying to—even if they hated studying it in school. Similarly, people who put together the network news shows quickly come to know a lot about social issues and current affairs as an incidental byproduct of the job they do. The best way to acquire knowledge like this is to participate in tasks that happen to require that knowledge. The best way to teach these subjects, therefore, is to provide environments in which the students can get involved in such tasks, and in which they can have ready, timely access to the knowledge they need.

In this paper we describe two computer-based incidental learning environments that we are building: One teaches United States geography by allowing the student to take simulated car trips around the country. The other teaches about social issues by allowing students to write and produce their own TV news program. In the sections that follow we will describe both programs and examine some of the general issues that must be faced when designing any incidental learning environment. In particular we consider the following three issues particularly crucial:

- **Integrating tasks and teaching goals:** How can the environment be structured to maximize the student's curiosity about the knowledge domain we wish to teach?
- **Guiding and coaching:** When should the program push the student to explore certain possibilities, and how can it do so?
- **Providing access to knowledge:** When the student's curiosity is excited by the task, how can the program make it easy for the student to find out something he wants to know?

ROAD TRIP: A Geography-Teaching Program

The Road Trip program is aimed at inner-city sixth-graders, many of whom may have little sense of where even the major U.S. cities and states are located. The program teaches United States geography by allowing the student to take simulated car trips around the country. When the student "arrives" at a particular city, he can watch video clips of interesting events that have occurred there.[1] Currently, there are videos of sports events, rock concerts, and natural wonders. Thus, from the student's point of view the program is an interface to television. The student can choose from a wide array of TV segments to watch, but in order to watch a segment the student must get to the location where the segment is set. If the student is a baseball fan, for instance, he may wish to watch Hank Aaron set the all-time career home-run record, which he hit in Atlanta. All he has to do to watch it is to figure out how to get to Atlanta.

The program consists of three components :

- **The Guidebook** contains descriptions of various regions and cities in the U.S., along with videos of interesting events. The student can browse the guidebook anytime to find out about events he might be interested in seeing, but can only view videos of things that took place at the student's current simulated location. The guidebook is a hypermedia document, with multiple organization schemes, allowing the student to search in different ways. So, for instance, the student can see all the events of a certain type (e.g., baseball events or concerts), or all the events in and information about a certain city, state or region.
- **The Locator Tool** consists of a set of hierarchically organized maps of the United States, which the student can use to locate chosen destinations. To start, for instance, the student is presented with a map of the entire U.S. The student can click on the region where he thinks his city might be. If he clicks on the wrong region, he is given a hint. (He is also given an opportunity to browse the section of the guidebook describing the incorrectly selected region. Rather than punish the student for incorrect guesses, the system uses this as an opportunity to advertise more destinations.) Once the correct region is selected, a detailed map of it is presented, and the student is allowed to select the appropriate state. This continues down to the level of clicking on the city itself. The student who prefers to can also use a map index to eliminate the trial and error.
- **The Travel Simulator** is the portion of the program that actually allows the student to take his trip. While the student is in the travel simulator, the main thing he sees is a map of the state he is currently in. The student always starts out in his home town. When in a city, the student can begin traveling by choosing any road that goes out of that city. As he travels, road signs appear, indicating roads he is crossing, cities he is passing, states he is entering, and points of interest. When the student gets off in a city, he can view the guidebook section on that city, and can view any or all of the videos set in that city. When the student passes a road sign, he is given the opportunity to view the section of the

guidebook which describes the relevant city or state. For instance, while traveling from Chicago to New York, the student will pass through Ohio. Upon entering Ohio, the program will encourage the student to go to the guidebook page that tells what there is to see there. If seeing Pete Rose break Ty Cobb's record interests the student then he can decide to go to Cincinnati before–or instead of–New York. It isn't really important that the student ever actually reach the original destination; what's important is that the student remain interested in seeking, discovering, and learning about new cities.

One principle guiding our design of incidental learning environments is that the student should not be forced into any one method of performing the task. Thus, when using Road Trip, the student can switch to any of the three components of the program at any time; different students are free to travel differently. Some may wish to start traveling right away without a set destination in mind. They may wish to just wander around looking for interesting places. Others may know what city they want to visit, but not know where it is; they may first use the locator tool to find the city, and then travel there, perhaps stopping off somewhere unexpected that they come to along the way. Many students will start without having any idea where they want to go, knowing only that they want to watch videos of, say, baseball. They may start by finding a fun video to see in the guidebook, then use the guidebook to find the location of the video, and then travel there.

Guiding and Coaching

Our belief that the student shouldn't be forced into a particular style of performing a task does not imply that we believe the student should be left to engage in completely unaided discovery. Burton and Brown (1982) demonstrate the pedagogical importance of guiding the student in a learning-by-doing environment. Both of our programs are capable of giving the student guidance and coaching. Road Trip's use of road signs and wrong guesses to advertise new destinations are examples of a kind of coaching. The videos themselves are also used to entice the student to visit new places. In addition to serving as a reward for reaching a destination, the videos make reference to events and characters that can be used to spark the student's interest in going other places. For instance, when watching the Pete Rose video, the student would be asked if he wanted to see a game in Detroit, where Ty Cobb played. The program uses every opportunity it can to suggest new destinations.

One idea we plan to implement, but have not yet, is to monitor where the student travels over successive sessions. If the student consistently ignores certain parts of the country, the program could stop advertising events in the regions that are well-traveled, and instead emphasize events in the regions ignored.

Broadcast News

The Broadcast News program teaches high school seniors about such topics as political science, international affairs, economics, and current events by allowing them to write and produce their own TV news show. Broadcast News supplies the student with newswire text and video feed (stored on video disc). This is the same source material that a local station would get, except that the student using the program uses material from a day in the recent past rather than the live feed for the current day.

During a session the student performs one or more of the main newsroom jobs:

- **The writer** writes the copy for a story and edits video to go along with the text;
- **The producer** chooses which stories the writer should work on, edits the writer's work, and sequences the various stories for the final broadcast;
- **The anchor** reads the stories the writer has written in the order determined by the producer.

The jobs not performed by the student are performed in predetermined fashion by the program. For instance, if the student plays producer then the program provides pre-written stories for the student, and pre-taped anchors to read the stories the way that the student decides they should be read. When editing a story, the student can click on any paragraph, and then delete that paragraph, modify the paragraph himself, or substitute an alternative paragraph submitted by the computer. When the student is done assembling his show, the program produces a video tape of that show, which the student can watch and save.

If the student needs additional background information in order to understand the issues related to the story he is putting together, he can use an on-line multimedia research library, which contains background stories from old news shows, as well as encyclopedia-like entries on the various people, countries, and institutions mentioned in the stories. A special section of the library includes evening news shows from the week prior to the current newsday.

Just as Road Trip permits students to travel in the style they choose, Broadcast News allows the students flexibility in how they can compile their show. Some students will choose to skim all of the news stories first, then make a "rough draft" of the show, and finally edit all of the stories. Others may prefer to study one story at a time, in-depth. The point is that the student is in control of the process, and thus is working using a methodology with which he feels comfortable.

Guiding and Coaching

Broadcast News contains expert commentary, stored on video disc, which it can present when relevant. Our experts include professional broadcast journalists, and political scientists. The student can "Ask the Experts" for advice about any of the decisions that he has to make, bringing up a video clip of one or more experts commenting on the student's choice. Sometimes an expert chimes in to point out errors or opportunities even when the student hasn't asked for advice. However, the experts are merely advisors, and don't always agree with each other. The student maintains final authority.

For example, when putting together a show about the events of September 21, 1990, a student might lead with a long story about anti-U.S. demonstrations in Baghdad. Our journalism expert would then point out that Arab-world demonstration videos flow into the newsroom all the time; such demonstrations are actually quite common, and therefore not as newsworthy as the student might have thought. The student must then decide whether to shorten the amount of on-air time spent discussing the demonstrations. The point is to link social and political issues we want to teach about to the journalistic decisions that the student needs to make.

Below are some major events that occurred on September 21, 1990, along with some associated issues the student must understand in order to produce a good newscast.

News Topic	Issues
Gulf Conflict: Bush says U.S. might strike first	President as Commander-in-Chief
	Middle East geography and politics
Gulf Conflict: Both sides expel some diplomats	Foreign affairs and the role of diplomats
Budget Summit between President and Congressional Leaders	Organization of U.S.government
	How a U.S. law is made
	Effects of budget on U.S. economy
Atlanta to Host 1996 Olympics	Economic impact of hosting a major event
	History, location, and socioeconomic makeup of Atlanta

Advantages of Incidental Learning Environments

Incidental learning environments offer a number of important advantages over more traditional teaching methods. By emphasizing an authentic task, rather than the assimilation of facts, incidental learning environments can motivate learning, situate it, and provide a context in which the student will be prepared to understand exactly why a teacher's remarks are relevant.

Learning is Motivated and Situated

Incidental Learning environments emphasize the fun and interesting uses of knowledge, thus motivating the student. Whereas traditional teaching methods often create a gulf between the acquisition of knowledge and its use, incidental learning environments such as the Road Trip and Broadcast News programs interleave the two so that the student is never tempted to ask, "Why would I want to learn that?"

There is an important distinction between using an artificial reward system to motivate the student, and providing an environment in which the student can engage in an *authentic* motivating task. In other words, it is not enough for the task provided by the incidental learning environment to be fun; it must be linked to the knowledge being taught in a way that genuinely illustrates why the knowledge is useful, and what is involved in using it. Miller and Gildea's (1987) experimental results point out the dangers of separating knowing from doing by showing that learning knowledge out of context does not ensure that knowledge will be stored in a usable form. Brown, Collins and Duguid's (1989) theory of situated cognition explains that the activity during which the knowledge is developed is an integral part of what is learned. The incidental learning systems presented here provide tasks that aren't only fun, they're also realistic uses of the knowledge being taught. Students use the geographic knowledge they learn to execute travel plans; they use knowledge of social issues to assess what is truly interesting and important about a particular day's events, and what is a fair way to describe those events.

In our opinion, the commercially successful Carmen Sandiego geography programs suffer from a failure to truly realize Brown, Collins, and Duguid's authenticity principle. The task environment does succeed in being motivating; students do enjoy chasing Carmen Sandiego around the world. However, geographical knowledge is not realistically linked to the task of locating the spy. Instead, the student is given a fun task with bits of geographical knowledge arbitrarily scattered through it. Like video games that interleave destroying monsters with math or vocabulary drills, the Carmen Sandiego approach succeeds at capturing the students' attention. However, whether or not it succeeds at imbuing the student with an appreciation of geography, or in teaching geographical knowledge in a form such that it will be available for later use, is an open question.

Just-in-time Teaching

Incidental learning environments provide contexts in which a teacher's comments and background articles can be presented at exactly the times when they address an issue that is active in the student's mind. This is the best way to ensure that the student will be interested in what the teacher or article has to say. It is also the best way to ensure that the student will store the information in a way that will make it available for future use (see Schank, 1990). For instance, it is one thing to sit in a class and hear a lecture about the separation of powers in the United States government, but it is quite another to attempt to put together a coherent story about the budget negotiations between the president and congressional leaders. When a student is struggling with this task, a teacher's explication of the respective powers of the congress and the president are welcome, and are immediately put to use.

Conclusion

Traditional teaching methods often force students to learn dull facts by rote memorization. Incidental learning environments, in contrast, teach by providing enjoyable, authentic tasks in which students actively participate. Learning is situated–students learn knowledge along with ways to use it, rather than sets of seemingly isolated facts. Our incidental learning environments allow the student significant control over the task, thus permitting him to learn in a comfortable and familiar style. The student should not be entirely unguided; expert commentary and background information are made available to the student at appropriate times, when the student is ready to use them. The combination of a motivating, situated task with timely teaching comments makes incidental learning environments powerful educational tools.

Notes

1. Both programs described in this paper are being developed on systems equipped with video jukeboxes, and special display cards that support on-screen video. The Broadcast News program configuration also includes a computer-controlled VCR and video camera.

Acknowledgements: We wish to acknowledge the members of the incidental learning project team, including Scott Dooley, Charles Earl, Ann Jacobs, Jarrett Knyal, Frank Luksa, Steve McGee, and Diane Schwartz. Bob Mulholland has provided valuable journalistic advice. We would also like to thank Roger Schank for his leadership.

This work is supported in part by a grant from the IBM Program for Improving K-12 Education through Innovative Uses of Information Technology, and by the Defense Advanced Research Projects Agency, monitored by the Air Force Office of Scientific Research under contract F49620-88-C-0058 and the Office of Naval Research under contract N00014-90-J-4117, by the Office of Naval Research under contract N00014-J-1987, and by the Air Force Office of Scientific Research under contract AFOSR-89-0493. The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner.

References

Brown, J.S., Collins, A., and Duguid, P. (1989). Situated cognition and the culture of learning. *Educational Researcher, 18*, 32-43.

Burton, R.R., and Brown, J.S. (1982). Investigation of computer coaching for informal learning activities. In Sleeman, D. and Brown, J.S. (Eds.), *Intelligent tutoring systems.* London: Academic Press.

Miller, G.A., and Gildea, P.M. (1987). How children learn words. *Scientific American, 257* (3), 94-99.

Schank, R.C. (1990). Teaching architectures. Institute for the Learning Sciences Technical Report #3, Northwestern University.

Point of View is Worth 80 IQ Points

Alan C. Kay
Apple Computer Inc.
131 S. Barrington Place, Suite 200, Los Angeles, CA 90049

Abstract: Historically, humans have extended themselves in two principle ways. First, by creating *amplifying tools*—physical and abstract structures that can be *manipulated*—such as the lever, wheel, telescope, language, writing, and mathematics. Second, by discovering how to get others to *absorb goals*—physical and abstract processes that can be *managed*—such as individuals, clubs, businesses, religions, countries, and cultures.

The metaphors of *manipulation* and *management* were immediately applied to human-computer communication in the late fifties when researchers first realized that the computer was a medium for extension and not just a calculation engine. The user interface—a simplifying metaphor theatrically portrayed—is the only "computer" most users ever see. Most of the successful designs so far have been of the manipulative amplifying tool type—Xerox PARC windows and pointing interface, spreadsheets, the Macintosh—but now there is an ever-increasing need for interfaces that can absorb goals and be managed.

In either case the key has been to direct communication to more than just the symbolic parts of the brain—to the kinesthetic (through pointing) and iconic (through pictures) mentalities as well. This has brought about the "Imaging Revolution in Learning" extending far beyond just making computers more friendly.

Though not much has yet been discovered about how human mentalities really function, the little that is known is both nonintuitive and crucial to the design process. Though we seem to have the same level of intelligence as thousands of years ago, our much more powerful contexts greatly amplify our ability to think. A user interface matched to human psychology provides a context for thinking and acting whose amplification can be truly remarkable.

As human and technological systems explode with capacity and complexity, the user interface designs of the future will be the principal path to simplicity and control.

On the Design of Novel Notations and Actions to Facilitate Thinking and Learning

Kenneth R. Koedinger
Department of Psychology
Carnegie Mellon University, Pittsburgh, PA 15213

Abstract: There are certain domains, for example, word problems or high school geometry, which even well-motivated students find particularly difficult. I claim that such domains are *enigmatic* to students because the problem solving notation used in current instruction does not provide a good reflection of the underlying problem solving processes necessary to be successful in these domains. I argue that some domains are not enigmatic in this sense, like algebra equation solving, and thus, are not good candidates for model-based notation design. However, domains that are enigmatic, like geometry theorem proving, can be made less so by (1) developing a cognitive model of skilled problem solving in the domain and (2) designing novel notations and actions that reify the underlying structures and processes of the model. In previous research, we found that skilled geometry problem solvers plan proofs much differently than the problem solving approach implied by textbook solution examples (and by previous cognitive models). A new cognitive model of this perceptually-based planning approach has been used to design novel geometry notations and actions that facilitate a way of thinking about geometry proofs that is more like the thinking of skilled problem solvers. These notations and actions have been implemented in a computer environment called ANGLE (A New Geometry Learning Environment).

Enigmatic Domains

There are certain subjects, for example, high school word problems and geometry proofs, which even well-motivated students find particularly difficult. I call such domains *enigmatic domains*. A key claim about them is that:

> Students find enigmatic domains particularly mysterious because the problem solving notation used in current instruction does not provide a good reflection of the underlying problem solving processes students must acquire to be successful.

While cognitive science tools can be used to identify these underlying problem solving processes, achieving effective instruction is not as simple as describing these processes or strategies to students. People are not very good at learning from explicit procedural instruction (e.g., consider trying to learn geometry problem solving by reading an AI program that solves geometry problems). Instead, people learn more easily from example solutions or cases (Anderson, Farrell, and Sauers, 1984; Riesbeck and Schank, 1989; Ward and Sweller, 1990; Zhu and Simon, 1987). However, not all examples are created equal: The ease of learning from example solutions depends crucially on the similarity between the example's notation and the problem solving process the student is supposed to learn. By developing a better understanding of these problem solving processes, particularly in enigmatic domains, we may be able to design notations that take some of the mystery out of acquiring enigmatic skills.

To illustrate the distinction between enigmatic and non-enigmatic domains, I contrast two domains: algebra equation solving and geometry theorem proving. This distinction is important because if the current notation is already a good reflection of the underlying problem solving process (i.e., in a non-enigmatic domain), designing better instruction may prove difficult. This appears to be the case for algebra equation solving.

A Non-Enigmatic Domain: Algebra Equation Solving

Field tests of two different intelligent tutoring systems for algebra have shown that while students clearly learn from these tutors, they do not appear to learn more effectively than students in a normal classroom (Lewis, 1989; Statz, Ormseth, McArthur, and Robyn, 1989). In addition, mathematics teachers report that algebra equation solving is not a particularly difficult skill for well-motivated students to acquire. It would be useful if we could characterize what it is about current instruction and/or the skill itself that makes learning algebra equation solving much more straight-forward for students than learning some other skills like geometry theorem proving.

One way to characterize the task of learning mathematics problem is in terms of two subtasks: 1) learning the basic problem solving operators, *operator application* and 2) learning how to choose among operators, *operator selection*. In algebra, both these subtasks can be accomplished using general weak methods–methods that most high school students have (at least) tacit knowledge of. Table 1 shows an example algebra solution.

Table 1
An Example Solution in the Domain of Algebra Equation Solving

(1)	$3x - 13$	$= 2(x - 3)$	
(2)	$3x - 13$	$= 2x - 6$	Distribute
(3)	$3x - 13 - 2x$	$= -6$	x's to left side
(4)	$3x - 2x$	$= -6 + 13$	Num's to right side
(5)	x	$= 7$	Simplify

Learning problem solving operators in this domain can be done in a straight-forward way by focussing on pairs of successive steps and inducing the operation that gets from one step to the next. For example, looking at the transition between steps 2 and 3 and similarly between steps 3 and 4, students can induce that terms, like $2x$ in step 2, can be moved from one side of an equation to another by switching the sign of the term. For example, in going from step 2 to 3, **2x** goes from the right hand side (RHS) in step 2 to **-2x** on the LHS in step 3, while in going from step 3 to 4, **-13** goes from the LHS to **+13** on the RHS.

Since many possible operators can apply to a particular step, students must learn to make good operator selections in addition to learning the operators themselves. Because of the nature of the algebra domain, students need to learn very little special-purpose operator selection knowledge. Instead, a the domain-general heuristic, *difference-reduction*, can be used to reliably select among operators. The difference-reduction heuristic is to choose operators that reduce (visible) differences between the current state and the goal state. (This heuristic is used, for example, as a key component of Newell and Simon's (1972) means-ends analysis method.) For example, from step 2 a student could notice that one of the differences between this state and the desired goal state (x = ?) is that there is an x-term on the RHS of step 2. This difference can be reduced by applying the "move-to-other-side" operator described above.

In algebra, the tasks of learning both operator application and operator selection are fairly straight-forward because the notation of example solutions facilitates the use of general problem solving and learning methods. That these to general methods can be straight-forwardly applied to learn algebra equation solving is evidenced by an early machine program that was successfully programmed to learn algebra in exactly this way (Neves, 1978).

An Enigmatic Domain: Geometry Theorem Proving

In contrast, the situation in geometry is much different. First, learning operator application is not so straight forward as it is in algebra. See Table 2 which shows an example solution from a geometry textbook. It is not easy to induce operators from successive steps in solution partly because successive steps are not always related, for example, step 4 comes from step 1 not step 3. The numbers in the parentheses on the reasons side of the proof can help make this connection but, in fact, examples in traditional textbooks often do not use these numbers. Even if the student makes the proper connections between steps, the job of inducing an operator that gets from say step 1 to step 4 is not easy. In fact, if the student only considers the statement notation itself, $\angle AED \cong \angle ECA$ and $\overline{AE} \cong \overline{AC}$, this induction is impossible. One must also consider the objects referred to by these statements (the angles and segments in $\triangle EAC$) in order to have any chance of inducing the **Opposite-Sides** operator (in a triangle, the sides opposite congruent angles are congruent).

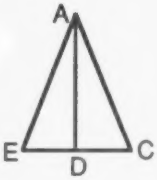
The task of learning operator selection is even more problematic. Domain-general heuristics like difference reduction are not effective in this notation. There is no natural way to characterize the differences, for example, between the givens $\angle AED \cong \angle ECA$ and **D midpoint** $\overline{EC}$ and the goal $\overline{DA}$ **bisects** $\angle CAE$ in a way that could aid in the selection of operators to reduce these differences.

Previous models of geometry problem solving (e.g., Gelernter, 1963; Anderson, Boyle, and Yost, 1985) have operator application knowledge that corresponds with the steps in two-column proofs, that is, the definitions, postulates, and theorems of geometry. Koedinger and Anderson (1990) called this problem space the *execution space* since it is made up of operators that correspond with the steps that problem solvers conventionally write down (or "execute") in solving problems. (The problem space discussed above for algebra is the execution space for that domain.) As further evidence that domain general search heuristics are not effective in this problem space, these previous computer models were only successful through the use of many domain specific heuristics.

Table 2
An Example Solution in the Domain of Geometry Theorem Proving

Given: $\angle AED \cong \angle ECA$
D midpoint $\overline{EC}$

Prove: $\overline{DA}$ bisects $\angle CAE$



PROOF:

Statements:	Reasons:
1. $\angle AED \cong \angle ECA$	1. Given
2. D midpoint $\overline{EC}$	2. Given
3. $\overline{DC} \cong \overline{ED}$	3. Def. of midpoint (2)
4. $\overline{AE} \cong \overline{AC}$	4. Opposite sides (1)
5. $\overline{AD} \cong \overline{AD}$	5. Reflexive (diagram)
6. $\triangle EAD \cong \triangle CAD$	6. Side-Side-Side (3, 4, 5)
7. $\angle CAD \cong \angle DAE$	7. Corresponding-parts (6)
8. $\overline{DA}$ bisects $\angle CAE$	8. Def. of bisector (7)

Identifying Implicit Planning

A straight-forward way to model problem solving in many domains is as a heuristic search in the execution space–the only trick is to find appropriate operator selection heuristics. From the perspective of a student, to the extent that the execution space provides a good characterization of skilled problem solving, as it seems to in algebra, his or her learning job is made easier. Execution operators can be induced fairly directly from the steps of worked out examples and may be supported by verbal descriptions in textbooks and lectures.

Although heuristic search in the execution space is a natural candidate for modeling problem solving in a domain, it may not be the problem space representation that skilled problem solvers typically use in this domain. It may be possible to find a better problem space representation in which solutions can be planned. Plans developed in this alternative representation can be executed in the conventional one. This is exactly what we have found that skilled geometry problem solvers do (Koedinger & Anderson, 1990). They plan key proof steps using a perceptually-based problem space representation (called the *diagram configuration space*) and deal with the details of the execution space after a complete (abstract) proof plan has been discovered. This approach yields a dramatic reduction in the geometry search space from more than 100,000 states in the execution space on a particular proof problem to just 12 in the diagram configuration space. In addition, the diagram configuration model (DC) provides a more accurate account of the planning behavior we have observed in the verbal reports of skilled problem solvers. Further details on the DC model can be found in Koedinger and Anderson (1990).

Essentially, we found that skilled subjects were skipping steps in the process of developing a proof plan. The regularity of this step-skipping within and between subjects suggested that subjects had "thinking steps" (operators) that are not represented in the execution space. From the perspective of the student, these thinking steps are an implicit part of the planning process which, in contrast to the execution operators, cannot be directly induced from worked out examples. In other words, when there is *implicit planning* in the thinking of skilled problem solvers, there may be aspects of a successful problem solving method which are hidden in the traditional curriculum.

Designing Interface Notations and Actions to Reify Implicit Planning

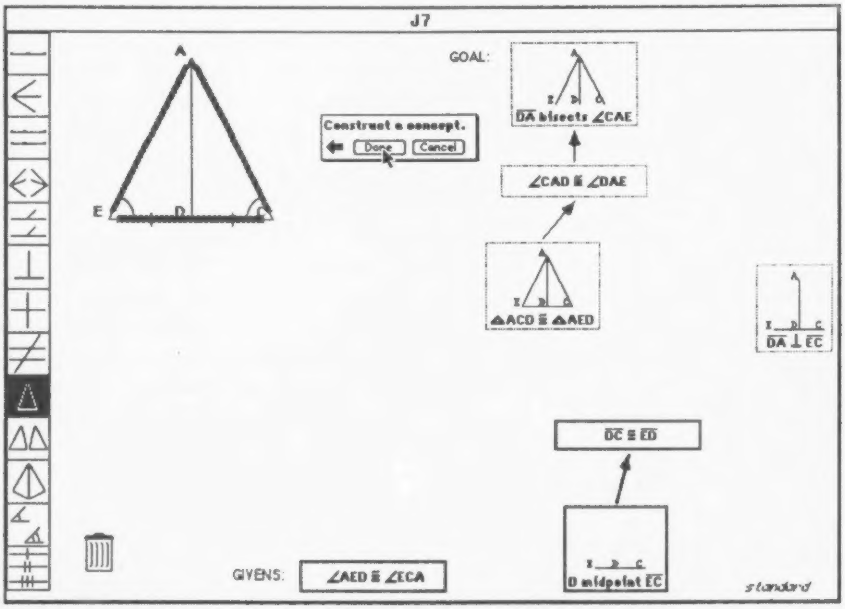
Building a cognitive model of implicit planning is the first step in this approach to model-based notation design proposed in this paper. Once an accurate model of implicit planning has been developed the challenge is to find a way to communicate this model to students. Since people seem to learn best by doing, directly communicating it to them does not usually help much. By design, this model contains problem solving process that are not reflected in the notation of the current curriculum. Thus, it may be possible to invent new notations which reify the previously hidden structures and processes. These notations can be the basis for interface design.

We have applied this approach in the design of ANGLE: A New Geometry Learning Environment. The cognitive model we developed, DC, played a key role in guiding the design process. Figure 1 shows the interface of ANGLE as it stands in the middle of the solution of the problem in Table 2.

An obvious design principle for a computer interface is that it should be easy to use and learn. A less obvious principle has to do with the role of interface as a subtle, yet ever-present, form of instruction. This implicit instruction comes both in the form of the *notations* used in the computer interface and also in the *actions* allowed by the interface. Notations should be created which mirror the important underlying representations of the problem solving model, while actions should be created which mirror the important underlying processes of the problem solving model. In this way, students can begin to internalize both the desired representations as they use the interface notations and the desired processes as they perform interface actions.

ANGLE Interface Notations

The ANGLE interface includes a number of examples of notations which reify representations in the cognitive model we developed. The key representations in the DC model are diagram configuration schemas that tie together perceptual knowledge of prototypical geometric categories with conceptual knowledge of properties and sufficiency conditions for these categories. The most prominent and important notational devices in ANGLE are the icons used for representing generic diagram configuration schemas and specific instances of these schemas. The generic schema categories are represented by icons in menu on the left in Figure 1. The icon for the ISOSCELES-TRIANGLE schema is currently highlighted (as are the segments in the diagram the make up an instance of this schema—discussed below). Student-selected schema instances are represented in the proof graph as a miniature picture of the relevant part of the diagram. For example, an instance the PERPENDICULAR-ADJACENT-ANGLES schema appears on the far right side of the screen and the miniature picture contains only the segments $\overline{DA}$ and $\overline{EC}$ that make up the perpendicular lines.



Construct a concept.

J7
A
E
D
C
GOAL:
$\overline{DA}$ bisects $\angle CAE$
Done
Cancel
$\angle CAD \cong \angle DAE$
$\triangle ACD \cong \triangle AED$
$\overline{DA} \perp \overline{EC}$
$\overline{DC} \cong \overline{ED}$
D midpoint $\overline{EC}$
GIVENS:
$\angle AED \cong \angle ECA$
standard

Figure 1. The ANGLE interface as it stands in the middle of the solution of the problem in Table 2. Students can work forward, backwards, or post "island" subgoals. An instance of the ISOSCELES-TRIANGLE schema is currently being instantiated in the problem diagram.

Following the Geometry Proof Tutor (Anderson, Boyle, and Yost, 1985), ANGLE incorporates a graph representation of proofs in contrast to the two-column format of traditional geometry instruction. This proof graph notation reifies the search process:

(1) By explicitly indicating how a correct solution must be a chain of steps linking the givens to the goal,
(2) By allowing the posting of subgoals as possible future links in the solution chain, and
(3) By explicitly indicating dead end solution attempts which are a common part of problem solving (even for experts).

In the proof graph proven statements appear in the bold boxes while unproven statements appear in the dotted boxes.

Following a conventional notation used on paper, ANGLE uses hash marks in the diagram to indicate segment and angle congruence statements that have been given or proven. For example, see the markings in diagram that indicate the angles in the given statement $\angle AED \cong \angle EDA$ are congruent. (The other proven statement $\overline{DC} \cong \overline{ED}$ is also marked in the diagram, but the markings are somewhat occluded by the highlighting.) These markings reify the equivalence class nature of segment and angle congruence in contrast to the binary relationships of the formal notation. In other words, to indicate that 3 angles are all congruent to each other using the hash mark notation, one can mark all three with the same marking–the marking serves as a token of the equivalence class containing all three angles. In contrast, the formal notation requires three binary statements to represent this situation, for example, $\angle 1 \cong \angle 2, \angle 2 \cong \angle 3$, and $\angle 1 \cong \angle 3$.

Finally, as a further aid to the acquisition of schemas, ANGLE highlights a schema within the problem diagram whenever the mouse passes over the corresponding schema instance icon in the proof graph. This is intended to reinforce the relationship between the schema and the rest of the diagram. This is a kind of dynamic notation which cannot be feasibly employed with paper and pencil.

ANGLE Interface Actions

Following the major processes in the cognitive model, ANGLE interface actions are broken down into (1) diagram parsing actions and (2) planning actions. The diagram parsing actions are those done in order to post schema statements. First the student selects a schema type and then indicates the lines within the diagram that make up an instance of this schema. Figure 1 shows a student instantiating the ISOSCELES-TRIANGLE schema–note that only the segments that make up this instance are selected, $\overline{AD}$ is not. This particular way of constructing a statement, as opposed to the way a student constructs one in the Geometry Proof Tutor or on paper, is meant to reinforce the relationship between the schema instance and the problem diagram in which it is embedded.

Planning actions are those done in order to justify schemas and their properties. To some extent these actions are more elaborate than those made in a two-column proof on paper: In ANGLE, the student must explicitly indicate the premises that lead to a conclusion (that is, by drawing the lines between them), while in the typical two column proof these links are only implicit. On the other hand, the planning actions are for the most part less elaborate than those required for a two column proof. Certain details required in a two column proof can be left out while constructing a plan. Students can omit a) certain statements usually required in a complete proof, for example, the reflexive statement $\overline{AD} \cong \overline{AD}$ can be omitted when the student goes to prove $\triangle ACD \cong \triangle AED$ in Figure 1, and b) the rules or "reasons" that usually appear in the right column of a two-column proof.

Any tutor interface (or notational scheme for that matter) is implicitly taking an instructional stance about what things are hard and/or important to learn and what things are not. Three important aspects of the instructional stance taken by ANGLE's interface are worth making explicit: (1) learning about the logical linkages between proof steps is hard and important, (2) learning the details of proof execution is less important and perhaps less hard than learning proof planning, and (3) learning how to parse geometry diagrams into particular chunks (DC schemas) is hard but it is important for successful search in a vast problem space. The first point is shared by the Geometry Proof Tutor's interface, but not by the two-column notation used on paper. The last two points are special to ANGLE's interface.

References

Anderson, J. R., Boyle, C. F. and Yost, G. (1985). The geometry tutor. In *Proceedings of the International Joint Conference on Artificial Intelligence-85*. Los Angeles: IJCAI.

Anderson, J. R., Farrell, R. and Sauers, R. (1984). Learning to program in LISP. *Cognitive Science*, 8: 87-120.

Gelernter, H. (1963). Realization of a geometry theorem proving machine. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and Thought*. New York: McGraw-Hill Book Company.

Koedinger, K. R. and Anderson, J. R. (1990). Abstract planning and perceptual chunks: Elements of expertise in geometry. *Cognitive Science*, 14: 511-550.

Lewis, M. W. (1989). Developing and evaluating the CMU algebra tutor: Tension between theoretically and pragmatically driven design. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Neves, D. M. (1978). A computer program that learns algebraic procedures. *Proceedings of the 2nd Conference on Computational Studies of Intelligence*, Toronto.

Newell, A. and Simon, H. A. (1972). *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall.

Riesbeck, C. and Schank, R. C. (1989). *Inside case-based reasoning*. Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Stasz, C., Ormseth, T., McArthur, D. and Robyn, A. (1989). An intelligent tutor for basic algebra: Perspectives on evaluation. Paper presented at the annual meeting of the American Educational Research Association, San Francisco.

Ward, M. and Sweller, J. (1990). Structuring effective worked examples. *Cognition and Instruction* 7(1): 1-39.

Zhu, X. and Simon, H. A. (1987). Learning mathematics from examples and by doing. *Cognition and Instruction* 4: 137-166.

Helping Teachers Teach Science Better: Case-Based Decision Aiding for Science Education

Janet L. Kolodner
College of Computing
Georgia Institute of Technology, Atlanta, GA 30332-0280

Abstract: AI-ED is a prototype for a case-based decision aiding system to help teachers teach elementary science better. The system holds the experiences of several experienced teachers and makes them available to other teachers as they are planning lessons. The effect is to make the intuitions and know-how of our best teachers concrete and available.

We read almost everyday about how students are getting bored with science or not learning enough of it. While the education establishment agrees that science should be taught in an experiential way, there is a large gap between the philosophy of such teaching and its practice. Teachers have not been equipped with the tools they need to teach experientially, and their general lack of scientific knowledge makes it particularly hard to teach science innovatively. Teachers are being asked to become facilitators of learning and to move from fact-teaching to approach-teaching, but without experience, it is hard to make the change. When is it appropriate, for example, to interact with students as they are discovering a new fact, and when should they be allowed to discover it for themselves? Our best teachers know the answers, but there is no good way right now of disseminating their intuitions and knowhow to others.

We are investigating the application of **case-based decision aiding** (Kolodner, in press) to helping elementary school teachers plan science lessons. This technology takes advantage of the fact that many solution to education problems already exist in the heads and experiences of our best teachers. Using case-based decision aiding, we are capturing this experience in order to put it in the hands of all teachers, making them more capable of teaching science in an exciting way and demonstrating the scientific method. Cases that are captured will, in effect, make concrete the intuitions of those teachers who are teaching science successfully.

Case-Based Decision Aiding

Case-based decision-aiding is based on a problem solving method called *case-based reasoning* (Hammond 1989; Kolodner 1988; Kolodner and Simpson 1989). In case-based reasoning, a reasoner makes decisions based on what worked and did not work in previous situations or "cases." There are several functions old cases or previous situations play. First, they can be used to suggest a means of solving a new problem. Second, they can warn of the potential for a problem to occur. Third, they can help in critiquing and repairing an already-worked-out plan.

Because case-based reasoning is like much of the natural problem solving people do, it provide us with a means for creating a new kind of decision-support system. A case-based decision aiding system stores a library of cases in its memory indexed by features that predict their usefulness. It works interactively with a human problem solver providing the human with cases to use in reasoning. In effect, it augments human memory with experiences that the human may not have had and provides a workbench for adaptation and evaluation, but it depends on the more flexible, creative reasoning of the human for decision making itself.

This technology seems particularly appropriate for helping teachers to plan exciting, engaging science lessons. While books and manuals give general guidelines, they do not tell teachers exactly what things to watch out for in the classroom. In general, we use our own experience and the experiences of others to help us figure out how to make guidelines work. This is what a case-based decision aiding system will provide for teachers.

Scenario of Operation

The system itself, called AI-ED, is in its infant stages right now. It has a library of science lesson plans derived from experiences reported in the literature and the experiences of the teachers we have interacted with. All lesson plans have an experiential emphasis. As teachers use the system, the library will be augmented with their experiences.

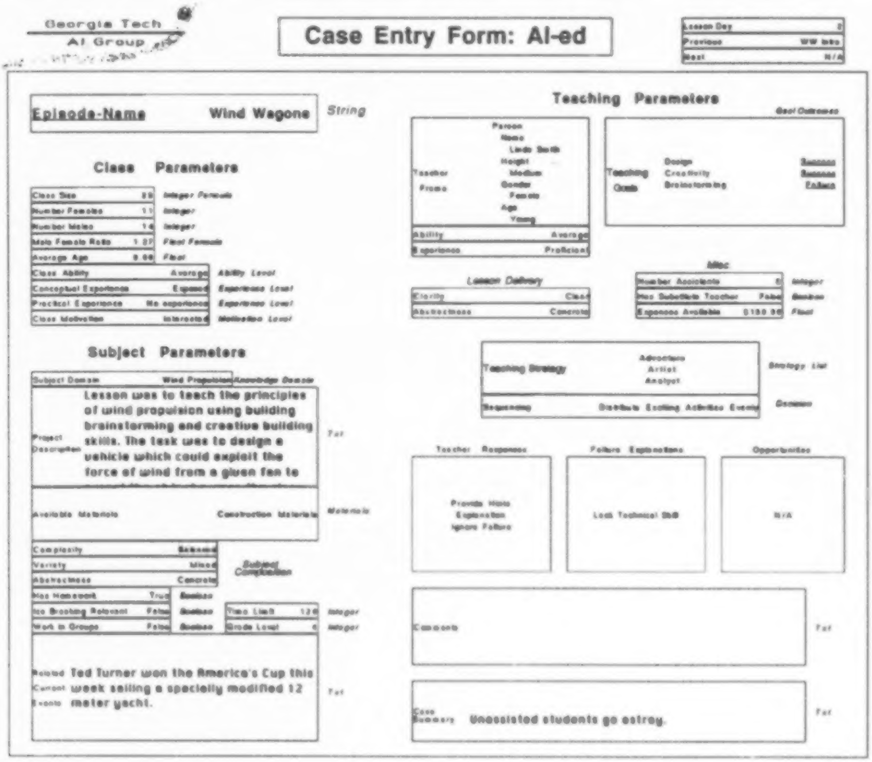
When complete, users will enter descriptions of subject matter, student population, available materials, and other relevant information to the system. The system will retrieve cases that can help the teacher formulate a lesson plan and evaluate it. Some plan and evaluate it. Some cases retrieved by the system will warn the teacher of things to watch out for; some will give guidelines for explaining, re-explaining, illustrating, and otherwise getting across difficult points, guidelines on what to emphasize, guidelines on kinds of interactions and the particular interactions they must have with students, etc. Some cases will help teachers anticipate areas of extra interest among the students, areas of confusion among the students, and student comments and questions that might arise and, in addition, provide suggestions for dealing with these situations. Other cases will help teachers determine the materials they need for a lesson and estimate the time necessary for preparation, conducting experiments, and providing explanations to students.

The lesson plan the teacher will create using the system might include a combination of group and individual activities, exploratory activities, and demonstrations–whatever is appropriate to the particular subject matter and student population. It will state needed materials and estimates of the time needed for preparation, conducting experiments, and explanations, and will provide them with explanations, illustrations, and examples they might be called on to give. The lesson plan will be a sort of contingency plan, circumstances that require following alternate branches of the plan and opportunities that might arise for adding something appropriate.

The current system, AI-ED, holds the experiences we collected from teaches at SciTrek (the Atlanta Science Museum) and the Boston Museum of Science. It is built using the Cognitive Systems CBR Schell, an intelligent database retrieval system. It currently hold 23 cases. A user can enter a new teaching situation and propose a lesson plan. The system recalls similar past cases and helps the user to debug his or her plan based on the outcomes of those cases.

In one situation, a user wants her students to explore gravitational force in a hands-on way by building parachutes. The lesson sequence she has in mind is to first show videos of different kinds of parachutes and their behavior, and then to have the children build parachutes and test them to explore gravity. The system recalls a class where students learned wind propulsion by destroying, building, and testing wind vehicles (shown in

Figure 1). Some students lacked the manual dexterity to build the vehicles they designed, others had trouble evaluating what was wrong with objects that would not be propelled. This case points out potential problems that the teacher should consider, i.e., that building should be a level students can handle and that some students will need help in evaluating the performance of their parachutes.



Case Entry Form: AI-ed

Lesson Day	2
Previous	WW Intro
Next	N/A

Episode-Name Wind Wagons String

Class Parameters

Class Size	25	Integer Formula
Number Females	11	Integer
Number Males	14	Integer
Male Female Ratio	1.27	Float Formula
Average Age	9.00	Float
Class Ability	Average	Ability Level
Conceptual Experience	Exposed	Experience Level
Practical Experience	No experience	Experience Level
Class Motivation	Interested	Motivation Level

Subject Parameters

Subject Domain: Wind Propulsion — Knowledge Domain

Project Description: Lesson was to teach the principles of wind propulsion using building brainstorming and creative building skills. The task was to design a vehicle which could exploit the force of wind from a given fan to — Txt

Available Materials: Construction Materials — Materials

Complexity	Balanced	Subject Complexion
Variety	Mixed	
Abstractness	Concrete	
Has Homework	True	Boolean
Is Brushing Relevant	False	Boolean
Work in Groups	False	Boolean

Time Limit	120	Integer
Grade Level	4	Integer

Related Current Events: Ted Turner won the America's Cup this week sailing a specially modified 12 meter yacht. — Txt

Teaching Parameters

Teacher Frame: Person — Name: Linda Smith; Height: Medium; Gender: Female; Age: Young

Ability	Average
Experience	Proficient

Teaching Goals — Goal Outcomes:

Design	Success
Creativity	Success
Brainstorming	Failure

Lesson Delivery

Clarity	Clear
Abstractness	Concrete

Misc

Number Assistants	0	Integer
Has Substitute Teacher	False	Boolean
Expenses Available	$150.00	Float

Teaching Strategy: Adventure, Artist, Analyst — Strategy List

Sequencing: Distribute Exciting Activities Evenly — Boolean

Teacher Responses: Provide Hints, Explanation, Ignore Failure

Failure Explanations: Lack Technical Skill

Opportunities: N/A

Comments — Txt

Case Summary: Unassisted students go astray. — Txt

Figure 1. The Windwagon Case

Issues and Approach

There are several problems that we are addressing as we aim toward an effective case-based decision aider for teachers.

Determining What Kinds of Cases Belong in the Case Library

Right now the system holds an ad-hoc set of cases. Making it truly useful will require us to do both a task analysis of the decision-making that works in preparing lesson plans and an analysis of the common decisions teachers make as they are putting lesson plans together. Since we want the system to help teachers make decisions that lead them to creating exciting engaging science lessons with appropriate content, we need to find out what kinds of lessons are effective, which kinds are not, and in what circumstances different kinds of lessons work.

Some teachers have reported to us, for example, that they need help with coming up with initial illustrative and motivating examples to catch student interest. To help teachers create illustrative examples, the case library needs to hold examples teachers have used for illustrative purposes and experiences teachers have had with those examples.

Representing Cases

Cases needed in lesson planning range in granularity from an individual interaction between the teacher and a student to sequences of lessons that together present a piece of the curriculum, and several things in the middle (e.g., an individual presentation). Each needs to accessed individually, but the connections between pieces must be maintained. In our current implementation, we represent only individual lesson plans (e.g., building and testing wind vehicles). However, we are aiming toward a system that holds libraries of lesson sequences, individual lessons, and individual interactions with students, and that maintains the proper connections between the parts.

In addition, each type of case requires different representational fields. Lesson sequences, for example, point to sequences of lessons that together achieve the lesson goals. The sequence of lessons and sequence of teacher subgoals that connect them are important to represent at this level. So is knowledge about background the students already have, since that might affect the sequences of lessons that teacher puts together. Particular lessons hold information about the materials needed, the questions that might arise, the problems that students had, and so on.

Some representational fields in AI-ED are supplied by the user, while others are calculated from those. Figure 1 shows a case from AI-ED, and shows the user-supplied fields in that case. Calculated fields are abstract descriptors of a case and are computed from its concrete features. The expected attentiveness of a class on some day might be a derived or calculated feature, as might the "mental gap" that needs to be bridged by the students (the distance between what they know and need to learn). In general, the set of important descriptive features that the system cannot derive by itself need to be input to the system, and the calculated features are derived from those. Figure 2 shows some of the fields that are calculated by AI-ED, and shows which explicit case fields they are calculated from.

Indexing Cases

The indexes of a case are the pointers or annotations that control the circumstances under which the case is retrieved. An approach to choosing indexes that has worked well in the past is to consider the circumstances in which each individual case could appropriately provide guidance and to index each case by features that describe those circumstances. It is the attention given to indexing that will insure that a case library that focuses on only a few concepts and teaching strategies will be more broadly applicable than only those concepts and strategies.

The indexing scheme, in effect, embodies the theory of similarity that a case-based system uses in retrieving cases. Cases indexed similarly are seen by the system as similar to each other. Thus, choice of case indexes is the key to making a system useful. Much careful work must be put into determining which features (both concrete and derived) will allow usefully similar cases to be retrieved.

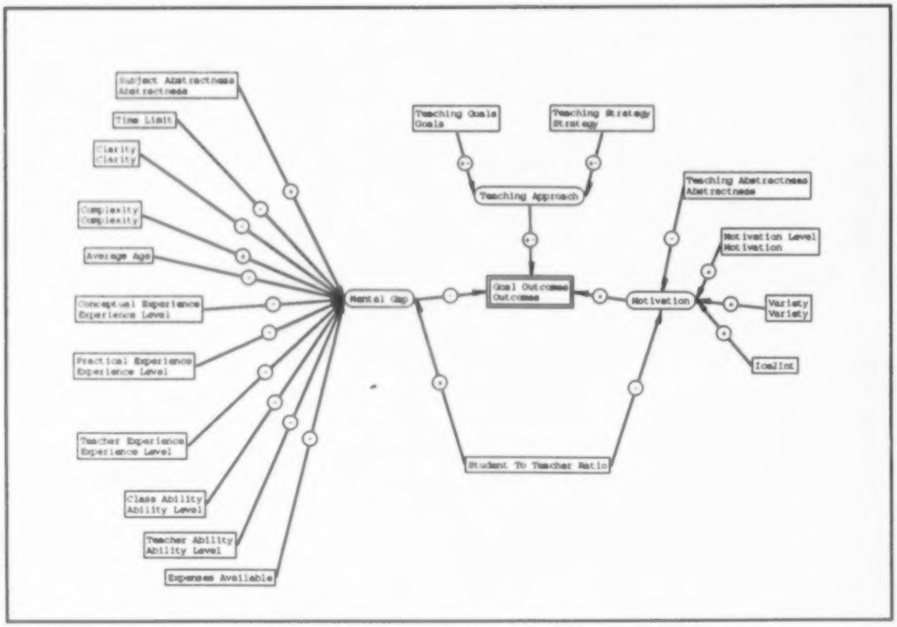

Subject Abstractness
Abstractness
Time Limit
Clarity
Clarity
Complexity
Complexity
Average Age
Conceptual Experience
Experience Level
Practical Experience
Experience Level
Teacher Experience
Experience Level
Class Ability
Ability Level
Teacher Ability
Ability Level
Expenses Available
Mental Gap
Teaching Goals
Goals
Teaching Strategy
Strategy
Teaching Approach
Goal Outcomes
Outcomes
Motivation
Teaching Abstractness
Abstractness
Motivation Level
Motivation
Variety
Variety
Ica2Int
Student To Teacher Ratio

Figure 2. Explicit and Calculated Fields

We have found two broad types of features useful for indexing teaching cases:

- teacher intentions (goals, plans, etc.)
- environmental features that effect the way teacher plans can be carried out.

We have been able to determine descriptors of teacher intentions by talking to teachers and examining teacher-training textbooks (Smith 1991). Cases at each level of granularity require slightly different indexes. Useful indexes for lesson sequences include several kinds of teacher goals (the facts to be taught, the skill/process to be taught), presentation method, teaching situation (e.g., introduction, review, reinforcement), and cognitive capability of the students. Individual presentations are indexed by style of presentation (e.g., teacher performs experiment, students perform designated experiment, students experiment in ad-hoc way) and design of presentation (e.g., comparison of different values on dependent variables, guided exploration).

The second type of feature for indexing is more ad-hoc. While teacher intentions give means of indexing standard solutions to problems, they do not give clues as to what the exceptions are that require different kinds of solutions. Sometimes, for example, class size makes a difference, other times attentiveness of students does. One teacher told usa story about how a school function the night before a lesson affected its outcome–the students had been up past their normal bedtimes and were too tired to concentrate. A partial set of environmental descriptors of cases includes class size, usual responsiveness of class, grade, age, experience of teacher with topic, and number of assistants available. Designating the full set of these indexes will require more interaction with teachers, collection of real cases that failed, and analysis of those failures.

These two classes of features for indexing are orthogonal to the classification set up in the previous section—concrete and derived features. In general, some features of each type are explicit (e.g., number of students) while others are derived (e.g., expected attentiveness of class). Indexes can be chosen from explicit or derived features. The important feature of an index is that it be predictive of something important in the case. AI-ED uses the CBRShell's method for indexing. Cases are indexed first by important derived features since derived features describe groups of similar cases features since derived features describe groups of similar cases whose individual features might vary considerably. When several cases with similar derived features are indexed similarly, they are further differentiated by their concrete features.

Closing

Educators reading this paper have no doubt noticed that we have not discussed using any theories of teaching or learning to guide our system development. The problem here is that theories that exist are generally too abstract to be of real help to teachers. Just as there is a gap between the philosophy of teaching science and the practice, so there is also a gap between theories of learning and development and the practice of teaching. As we collect cases, however, we are trying to make sure that the teaching strategies embedded in cases in the system are consistent with what is known about how people learn.

At the same time, we see this lack in the education literature as an opportunity for the cognitive science community and for our project. As we talk to teachers and collect and analyze cases, we will gain insights about teaching strategies and their applicability, learning by exploration and discovery, teaching the scientific method (as opposed to facts), cognitive capabilities and cognitive development of elementary school students, the kinds of examples that foster learning, differences between lessons for highly-enriched and poorly-enriched students, and so on. We are hoping to be able to cull effective teaching strategies from the many cases we encounter and to express them at a level of detail appropriate for teachers to use in creating lesson plans.

There are other issues that must be addressed too in building the aiding system that can be put into the classrooms. Human-computer interaction issues will need to be addressed, the effectiveness and ease of use of the system will have to be evaluated, and several additional CBR issues will have to be addressed.

While most AI education projects concentrate on providing instruction to students, we are emphasizing aiding of teachers. As good as automated systems become, the human teacher will still be needed for at least some tasks. In this project, we have just begun to explore how case-based decision aiding can help teachers to prepare an environment for students that is conducive to learning science. Hopefully, what we learn in this project can productively be put to use in creating other systems to help teachers facilitate learning of many different kinds.

Acknowledgements. This research is supported in part by IBM through the University Grants Program and in part by DARPA under contract No. F49620-88-C0058 monitored by AFOSR. The implementation of AI-ED was done primarily by Mike Cox with help from Sam Bhatta and John Barber.

References

Hammond, K. (1989). *Case-based planning*. New York: Academic Press.

Kolodner, J.L. (1988). *Proceedings: Case-based reasoning workshop*. San Mateo, CA: Morgan-Kaufmann.

Kolodner, J.L. and Simpson, R.L. (1989). The MEDIATOR: Analysis of an Early Case-Based Problem Solver. *Cognitive Science* 13(4): 507-550.

Kolodner, J.L. (in press). Improving human decision making through case-based decision aiding. *Artificial Intelligence Magazine*.

Smith, D. (1991). Principles for case representation in a case-based aiding system for lesson planning. In *Proceedings: Third DARPA Case-Based Reasoning Workshop*. San Mateo, CA: Morgan-Kaufmann.

Developing and Evaluating Novel Strategic Computer Simulations: A Case Study in Translating an Instructional Pedagogy to the Classroom

John A. Kruper
Division of Biological Sciences
University of Chicago, Chicago, IL 60637

Eric M. Nelson
Academic & Public Computing
University of Chicago, Chicago, IL 60637

Abstract: The goal of this study was to investigate the utility of an educational pedagogy that uses strategic computer simulations to allow students to actively construct meaning by defining problems, constructing and testing alternative hypotheses, and communicating subsequent evaluation of these hypotheses to peers.

Using this pedagogical framework, our research group developed and is presently testing **Biota**. **Biota** is a simulation tool for studying the processes which influence the sizes of plant and animal populations.

In this report we describe **Biota** and how its design reflects the chosen pedagogy. Next, we present the results of a pilot study that used **Biota** in an introductory biology laboratory unit on competition. The results showed that, while learning outcomes did not differ significantly between groups, the learning processes of the groups did differ. These results add to previous findings which suggest that, while not necessarily improving problem-solving *performance*, strategic simulations can offer students valuable *experiences* to develop deeper content understanding.

The BioQUEST Pedagogy and Strategic Simulations

A growing movement in science education stresses helping students develop an understanding of scientific principles and then generalizing these principles through experiential problem solving (Linn, 1986; Duschl, 1990; National Research Council, 1990). This view was elegantly expressed by the Committee on Science, Education, Engineering and Public Policy (1984) which said that "in addition to traditional competencies, students increasingly should learn the thinking skills to manage information, formulate effective probing questions, test hypotheses, make judgements, express themselves logically and lucidly, and solve problems."

One group that has actively embraced this approach to science education is a consortium of educators and researchers known as Project BioQUEST [1]. Its members see the practice of science as a design-like activity whose essence is open-ended, effect-to-cause problem solving (Peterson *et al.*, 1987). This view is in marked contrast to the static, cause-to-effect problem solving activities traditionally used in our classrooms and scientific textbooks.

To provide students direct experience in the strategic, effect-to-cause process of *doing* science, BioQUEST software uses *computer microworlds*. These microworlds provide a scientifically accurate model (or group of models) of a complex system along with a set of realistic research tools with which to investigate the system. When we exploit the power offered by the computer, we can create *strategic simulations* which possess the following characteristics (Jungck & Calley, 1985):

(1) Novelty of problems each time a program is run
(2) Realistic outcomes for each experiment performed
(3) Infinite opportunities to perform experiments
(4) Computational power
(5) Speed in obtaining results
(6) Large data size
(7) Facilitate successive hypothesizing and both logical and numerical testing
(8) Sequentially developed difficulty involving an increased quantity of natural phenomena

In the BioQUEST pedagogy, these software attributes are used to focus the student's experience on three elements of scientific practice (Peterson & Jungck, 1988):

(1) **problem posing:** the act of composing problems;
(2) **problem solving:** the activities associated with working toward solutions to the posed problems; and
(3) **persuasion of peers:** the act of going public with a solution.

To members of BioQUEST, these "3P's" are as important to teaching and learning science as the 3R's were to education at the turn of the century.

The combination of a rich technology with robust philosophy results in a revolutionary reorganization of the science classroom environment. The teacher is no longer a repository of catalogued knowledge, but rather functions as an expert strategist who actively collaborates with student research groups working on open-ended problems (Jungck & Calley, 1985). This notion is emphasized by a software design standard which establishes a separation between the construction of a particular simulation, and the use of that simulation as a "black box" that presents a problem to be studied. In the BioQUEST framework, the more these programs can operate like real labs, the better the student will understand the process of science, and thus understand the very rich and complex context of what are loosely termed scientific "facts."

A Case Study in Instructional Software Design: Population Biology and Biota

The University of Chicago is a research university with a strong belief that beyond a base of knowledge, it must also teach the intellectual and practical skills needed to perform research and extend understanding. Such an environment is naturally hospitable to the 3P's approach, so it is not surprising that a diverse group was attracted to developing **Biota** as a strategic computer simulation in population biology[2]. The program's design grew from intense collaboration among the design team, which drew heavily from members' back-grounds in education, cognitive psychology, philosophy and computer science as well as population biology.

Biota simulates empirical data-generation and field data-collection experiments in population dynamics. Specifically, it models the population dynamics of single and multiple species in a common region, the migration of populations among those regions, and simulates the properties of experimental tools such as traps, transects and quadrats. It allows users to explore a variety of questions concerning the influence on population dynamics of factors such as birth and death rates, density effects, types of growth (e.g. exponential, logarithmic), multiple simultaneous species relationships (e.g., A preys on B, B is host to parasite C, etc.), and migration (among others). Users can also explore the influences on data of different experimental designs, research tools, and sampling methods.

Design of Biota

Biota has been designed to meet the requirements of a strategic simulation. There are two primary modes: one is a 'glass box' in which simulations can be constructed and explored; the other, a 'black box' problem mode where experimental methods and tools are used to explore the underlying simulation. Except for machine-specific memory limitations, the ability to create complex simulations and field studies is unconstrained (#1 above). The results of experiments reflect both the statistical properties of the tools and methods, and the adequacy of the experimental design (#2). The same problem can be examined using many experimental designs, allowing the student to quickly compare the consequences of different designs (#3). The modelling algorithms attempt to maximize speed and minimize memory requirements. [Even on the low-end target machines (Mac+ with System 6.0 & 1Meg RAM), we have found that fairly complex simulations (e.g., 3 species, 4 regions) run with acceptable speed (i.e., you rarely see the busy icon) (#4 & #5).] There are no intrinsic limits to the amount of data that can be collected (#6). Simulations and problems can run simultaneously, allowing direct comparison of results. A notepad is available for recording observations, conclusions, etc. (#7). Simulations can be as small as one species in one region; new species and regions can be incrementally added, and properties, such as types of relations, growth patterns, population levels and migration can be varied within and between simulations (#8).

Our implementation reflects a number of design principles:

(1) The interface should be as concrete as possible. Where problem-solving requires an understanding of visual or spatial factors, the interface should provide it.
(2) Clarify, don't simplify.
(3) Structure should facilitate, but not constrain.
(4) Implement the general solution first, then provide tools for special cases and shortcuts.

Early on, we chose to construct a general system in which simulations could be flexibly constructed, rather than provide a number of preset, or randomly generated, simulations that posed the "interesting questions." We chose this direction for two main reasons:

Pragmatics: Any modelling engine general enough to handle a broad range of interesting problems would require many design/test cycles of each problem. We felt that defining those problems and finding the right parameters would take as much time as building an interface that lets the user construct their own simulations.

Philosophy: In a domain as complex as population dynamics, model construction is as much a problem-solving activity as is experimentation. To make this unavailable, or inflexible, would violate a basic aspect of the BioQUEST philosophy. Further, the program designers' preconceptions should not define what is an 'interesting problem' or the means for solving it. The program should not contain constraints on use of its features, except as they reflect real life constraints. Structural aspects of the program (window layout, order of tools in an set, etc.) can provide implicit guidance in developing a simulation, designing a study of a problem, or in the interpretation of results, but they must not impede other approaches to problem solving.

Implementation of Biota

For both practical and strategic reasons, we chose to develop **Biota** on top of Apple's MacApp object library, using Object Pascal as our programming language.

Modelling

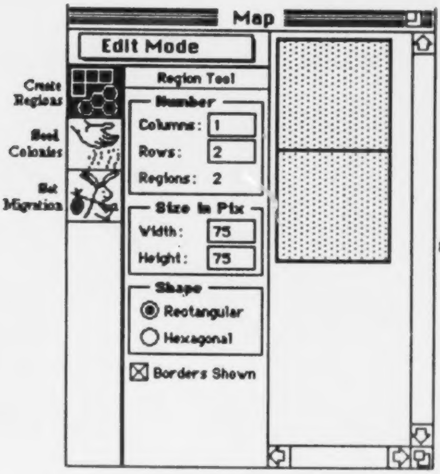
The design of the modelling engine is the core of **Biota's** flexibility. Using this general approach, we can treat the relations between species (e.g. parasite/host) the same way we treat the consumption of resources (e.g. water). The underlying object-oriented architecture is hospitable to the inclusion of new types of relations and mathematical models, including, in the future, the option to specify new models. Thus, a user can model any logical combination of relations among any number of species or resources, using any of the available mathematical models. This generality even allows **Biota** to be used to explore actual field data.

Simulation Mode and Problem Mode

Biota's *Simulation mode* permits the user to examine 'What if' scenarios by manipulating the underlying parameters of the models. From such simulations, an instructor can create problems that focus on specific features of population dynamics, research methods or data analysis. When a simulation is converted to a problem, it can only be examined subsequently in *Problem mode*: users only have access to research tools, there is no access to the underlying parameters.

Figure 1a shows the arrangement of tools available in Simulation mode, where the construction of the map, distribution of populations, and the specification of migration routes take place. Figure 1b shows the tools available in Problem mode, where sampling and some experimental manipulations can be done. Particular care has been taken in the simulation of sampling methods, so that they give statistically realistic data.

Once a simulation or experiment has been specified, the user starts the program running and the 'Summary Window' (Figure 2) appears. This provides simultaneous graphical and tabular views of the data. Simulation controls are in the upper right. Data generated can be saved in a text file for subsequent analysis using statistical software, or for incorporation into a word processing document. In addition, the Summary Window provides access to the notebook. The ability to interactively control the simulation, save and/or export data, and record experimental observations within the simulation document, all represent common BioQUEST design features.


Map
Edit Mode
Create Regions
Seed Colonies
Set Migration
Region Tool

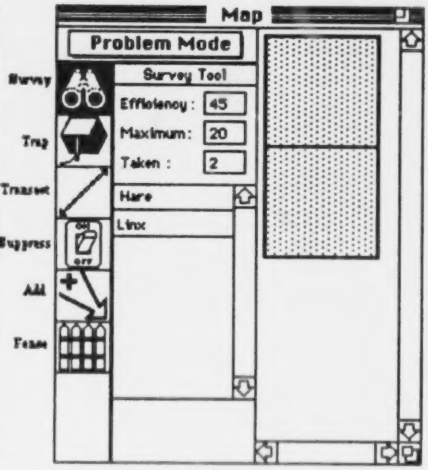
Number
Columns: 1
Rows: 2
Regions: 2

Size in Pix
Width: 75
Height: 75

Shape
● Rectangular
○ Hexagonal

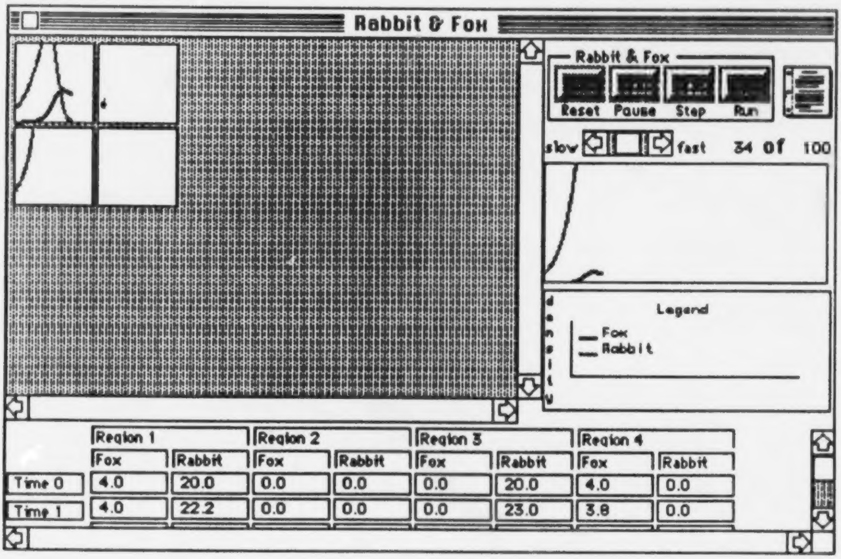
☒ Borders Shown

Figure 1a: Simulation construction


Map
Problem Mode
Survey
Trap
Transect
Suppress
Add
Fence
Survey Tool
Hare
Linx

Efficiency: 45
Maximum: 20
Taken: 2

Figure 1b. Problem mode


Rabbit & Fox
Rabbit & Fox
Reset
Pause
Step
Run
slow
fast
34 of 100
density
Legend
Fox
Rabbit

	Region 1		Region 2		Region 3		Region 4	
	Fox	Rabbit	Fox	Rabbit	Fox	Rabbit	Fox	Rabbit
Time 0	4.0	20.0	0.0	0.0	0.0	20.0	4.0	0.0
Time 1	4.0	22.2	0.0	0.0	0.0	23.0	3.8	0.0

Figure 2. Summary window

Classroom Study

Research into Computer Assisted Instruction (CAI) in science education has involved: measuring and comparing learning outcomes of students taught using CAI materials versus control groups (Choi and Gennaro, 1987; Friedler *et al.*, 1990); studying and characterizing novices' and experts' scientific reasoning processes (Slack and Stewart, 1990); and identifying, and possibly correcting, student misconceptions (Kinnear, 1983; Browning and Lehman, 1988).

These efforts provide insight into effective use of CAI. In particular, they note that if attention is only focused on the "final product" (i.e., learning outcomes), then CAI is often no better than other traditional instructional approaches. Rather, the advantage of CAI lies in enabling students to actively engage in the *process* of scientific discovery.

With this contrast in mind, we initiated a pilot study of **Biota** to examine its effectiveness as a curricular tool. Students in an introductory population biology class at the University of Illinois at Chicago were drawn from three laboratory sections containing approximately 32 students per section. Each section participated in a 50-minute pre-lab discussion where a pre-test was administered and the upcoming laboratory discussed. This was followed by a 2-hour laboratory session designed around the subject of competition as originally formulated by Gause (1932). In this setting, students followed either a "wet-lab" protocol or analogous **Biota** problem-mode or simulation-mode protocols. A post-test was given to all students one week after the laboratory session.

In designing our study we wished to answer two primary questions:

(1) How does the laboratory *learning environment* compare between students conducting a traditional 'wet' laboratory versus those completing an analogous study using **Biota**?

(2) How do *scientific reasoning skills* compare between students conducting a traditional wet laboratory, students conducting an analogous **Biota** laboratory operating only in Simulation Mode, and students conducting an analogous Biota laboratory operating only in Problem Mode?

As space limitations prevent us from presenting the full results, we will briefly summarize findings most relevant to the above research questions. A fuller presentation of these results is in preparation and will be published elsewhere.

To answer Question 1, we examined the pattern of questions asked by students during the wet lab and the **Biota** labs. We classified these questions into three types: 1) those concerned with the *mechanics* of the experimental tools employed, 2) those concerned with the *methods* of identifying, collecting, and tabulating data, and 3) those concerned with the *interpretation* of the data and the formation of further hypotheses.

We found that the absolute number of questions asked by the students in each lab were much higher in the wet lab (wet lab, 104; simulation, 29; problem, 22), but in all labs the relative distribution of questions across the categories was not substantially different[3]. Furthermore, we note that in the same time it took the wet labs to set up and do their initial counts of yeast (i.e., data collection), the two **Biota** labs were able to work through all phases of experimentation, including data collection, analysis, and interpretation *several times*. Thus, students in the **Biota** labs were indeed practising Problem Posing and Problem Solving.

To answer Question 2, we examined performance on pre- and post-tests which required students to analyze data generated from experiments similar to those performed in the lab (although not necessarily displaying the same competitive outcome), and to predict the outcome of experiments conducted under different initial conditions. With each, students were asked to posit a set of factors that could account for their observed/predicted data.

The results show that, with respect to the ability to correctly analyze data or to predict the shape of new data, there were no significant differences either over time within groups, or between groups at the same time. This lack of change can be attributed to a number of reasons ranging from instrument design to content complexity, but we feel the primary agent is the brevity of a single exposure in a 2-hour lab. Consistent with CAI findings mentioned above, it may be that strategic simulations are most effective when they are part of progressively deeper study of a domain. Simulations readily lend themselves to this approach, because they permit the modelling of increasingly complex phenomena, and the student is not so tied to a laboratory containing sophisticated biologicals or other expendables.

Summary

We are not suggesting that computer simulations based on the 3P's can, or should, replace regular laboratories. Science education must be founded on direct experience with the subject of study. But commonly, the focus of the wet lab stays on the mechanics of working the instruments and figuring out just 'what that organism looks like under the microscope'. Further, to make the experience more 'relevant', a cause-to-effect explanation such as "We are looking at this because it is an example of theory X" is typically offered. What is lost in this approach is how a single lab exercise would constitute only a small part in the overall design of a larger experiment, whose questions are much more interesting because their answers are not known. In strategic simulations we have the complement to the wet lab; here students see the effects, and must infer the cause(s). In solving these problems, the mechanics of the lab become secondary, and the conduct of a research plan can be brought into focus. Indeed, we suggest that this long-term aspect of strategic inquiry may be the crucial factor in enhancing student problem solving--we certainly know that even with the capacity to complete a full investigation, a 2-hour session does not suffice However, the computer easily provides just this capacity, and it is to this approach we are currently turning our attention.

Notes

1. Further information regarding the BioQUEST (Quality Undergraduate Educational Simulations and Tools in Biology) program can be obtained from Dr. John Jungck, Department of Biology, Beloit College, Beloit, WI 53511.
2. Members of the Biota design team are: Jim Danbury, Ben Jones, John Kruper, Jim Lichtenstein, Eric Nelson, Jeff Schank, Bill Sterner, Joyce Weil, and William Wimsatt.
3. Given that all protocols were equally detailed, we feel this indicates that **Biota's** interface is intuitive and accommodates novice user exploration.

Acknowledgements: The authors wish to acknowledge the many contributions made by the Biota development team in making this study possible.

References

Browning, M. E. & Lehman, J. D. (1988). Identification of Student Misconceptions in Genetics Problem Solving Via Computer Program. *Journal of Research in Science Teaching, 25*(9), 747-761.

Choi, B. & Gennaro, E. (1987). The Effectiveness of Using Computer Simulated Experiments on Junior High Students' Understanding of the Volume Displacement Concept. *Journal of Research in Science Teaching, 24*(6), 539-552.

Committee on Science, Education and Public Policy, the National Academy of Sciences, the National Academy of Engineering, and the Institute of Medicine (1984). A report of the research briefing panel on information technology in precollege education. In *New Pathways in Science and Technology: Collected Research Briefings (1982-1984)* New York: Vintage Books.

Duschl, R. A. (1990). *Restructuring Science Education: The Importance of Theories.* New York: Teachers College Press.

Friedler, Y., Nachimias, R., & Linn, M. C. (1990). Learning Scientific Reasoning Skills in Microcomputer-Based Laboratories. *Journal of Research in Science Teaching, 27*(2), 173-191.

Gause, G. F. (1932). Experimental Studies on the Struggle for Existence. I. Mixed Population of Two Species of Yeast. *Journal of Experimental Biology, 9,* 389-402.

Jungck, J. R. & Calley, J. N. (1985). Strategic Simulations and Post-Socratic Pedagogy: Constructing Software to Develop Long Term Inference Through Experimental Inquiry. *American Biology Teacher, 47*(1), 11-15.

Kinnear, J. F. (1983). Identification of Misconceptions in Genetics and the Use of Computer Simulations in Their Correction. In H. Helms & J. D. Novak (Ed.), *International Seminar on Misconceptions in Science and Mathematics,* Cornell University, Ithaca, New York: Cornell University Press.

Klahr, D., & Dunbar, K. (1988). Dual Space Search During Scientific Reasoning. *Cognitive Science, 12,* 1-48.

Linn, M. (1986). *Establishing a Research Base for Science Education: Challenges, Trends, and Recommendations.* Washington: National Science Foundation.

National Research Council (1990). *Fulfilling the Promise: Biology Education in the Nation's Schools.* Washington: National Academy Press.

Peterson, N. S. & Jungck, J. R. (1988, March/April). Problem-Posing, Problem-Solving and Persuation in Biology Education. *Academic Computing,* p. 14-17.

Peterson, N. S., Jungck, J. R., Sharpe, D. M., & Finzer, W. F. (1987). A Design Approach to Science - Simulated Laboratories: Learning Via the Construction of Meaning. *Machine-Mediated Learning, 2*(1&2), 111-127.

Slack, S. J. & Stewart, J. (1990). High School Students' Problem-Solving Performance on Realistic Genetics Problems. *Journal of Research on Science Teaching, 27*(1), 55-67.

Organizational Memory as a Support for Learning and Performance: Prototypes and Issues

James M. Laffey
Apple Computer, Inc., Apple Training & Support Readiness
MS 72Z, 20525 Mariani Ave, Cupertino, CA 95025

N. Rao Machiraju
Apple Computer, Inc., Apple University
MS 83U, 20525 Mariani Ave, Cupertino, CA 95025

Ravinder Chandhok
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA

Abstract: We have developed an integrated support and learning system (IS&LS) to augment the way people build and gain access to organizational memory. The system we are building is based on the belief that much of the competencies needed to be an effective knowledge worker are learned on the job. This learning is supported by access to organizational memory while in the process of doing one's job. An integrated support and learning system can facilitate learning by helping knowledge workers build, store, dynamically access, and interact with organizational memory.

This paper presents organizational memory as a support to learning and performance, describes the development of a prototype system called "NNAble," and discusses issues raised during our development process.

Introduction

Lave (1988, p. 170) asks "What would happen to theorizing about cognition if investigators were moved to the sites of the activity whose interpretation was under debate?" As learning scientists and practitioners we have restated this question to be, "What would be the impact on competency and competency building if we supported learning in the work environment, rather than thinking of learning and working as separated in time, place and structure?" Guided by this question we have studied a group of knowledge workers and identified the knowledge they use to solve problems and support their performance. We have adopted the term "organizational memory" to describe this knowledge.

Nearly 30 years ago Douglas Engelbart provided a vision for augmenting the human intellect. By augmenting he meant "increasing the capability of a man to approach a complex problem situation, to gain comprehension to suit his particular needs, and to derive solutions to the problem." (Engelbart, 1962, p. 1). Critical to Engelbart's vision is a framework which leads us to consider the worker, the learner, and the work situation as an integrated whole with conditions that negate or facilitate "increasing capability."

Discussions of situated cognition (Collins, Brown & Newman, 1989), cognitive apprenticeships (Brown, Collins & Duguid, 1989), contextualized learning (Fischer, Lemke & McCall, 1990), generative learning (Wittrock, 1977), self-regulation of knowledge internalization (Iran-Nejad, 1990), and reflective practitioners (Schon, 1987) point to a view of learning or competency building as a multi-source, integrated, situated and goal directed process.

Drawing upon these notions, we have set out to build an integrated support and learning system that relies on organizational memory for augmenting the worker or authentic performer rather than through abstracted instructional practices. Organizational memory refers to the knowledge that an organization has or could have about its business. It includes what is archived in electronic and paper documents, but even more critically, what people in doing their work have learned about policies, practices and solutions. Much of organizational memory is knowledge situated in the activity and context in which it was produced. Augmenting the worker will mean shaping the work environment in several ways: supporting exploration; facilitating the making, articulating and testing of hypotheses; tailoring training excursions to the work task; and connecting the worker to a knowledge base of the expertise used by practitioners that is contextually relevant (i.e., filtered by the work task and situation).

Additionally, this direction is also supported by a view of the changing work place and work demands. Knowledge workers of today have to act in the face of increasing uncertainty and ambiguity and must draw upon distributed expertise to solve problems (McHale, 1983). Organizations are in continual transformation and new systems for learning are critical for supporting the ever changing competency needs of the knowledge worker.

Organizational Memory

As a first step towards building IS&LS, we have studied how Apple Technical Coordinators (or ATC's, a sub-set of Apple knowledge workers) acquire, utilize and transmit problem solving knowledge in the context of doing their jobs. At Apple, ATC's are chartered with trouble-shooting computer and communications systems used by Apple employees. For example, if your workstation suddenly couldn't print documents properly you would call your ATC. At that point, the ATC is charged with "getting it fixed". We found a large variance in the trouble-shooting capability from one ATC to the next. Despite this variance, however, there were common elements present in most of the trouble-shooting we observed. We found that ATC's rely on four key support elements during problem solving sessions.

The first support element is the use of context rich cases. Knowledge workers often explain their actions in a new situation by referring to processes and outcomes of a prior analogous situation. They often try to help their colleagues by replaying cases or stories from their own experiences.

A second source of support is access to collaborators. There are many and varied collaborative roles, but it is clear that most problem solving and indeed most successful performances are dependent upon some collaborative interventions. Three of the key roles of a collaborator are to serve as an agent for finding information, to explore an alternative perspective and to help a person practice/rehearse a solution.

The third source of support is access to archived resources. Things which are part of your personal experience which are available to you through some search strategy (e.g., an article that you read a month ago that you remember how to find). These resources also include items like technical reference libraries, or printed documentation.

The fourth source for self-directed problem solving is through practice (trial and error and/or modeling), wherein people try performing tasks or watch other people perform them. These approaches are often combined in that the person has watched or heard of someone doing a procedure and now tries it out in the present situation.

Integrating these four elements and creating dynamic updating and access would constitute an organizational memory for ATC's. It is important to note that these sources of support are not orthogonal. The "doer" almost always uses multiple components and integrates them to address the problem at hand.

Lack of organizational memory means the folklore that results from hearing about previous cases is not widely available–the stories are distributed ad hoc through chance meetings or informal messages. Appropriate collaborators are often only known within a small group and the knowledge of "who knows what" tends to be scattered and uneven throughout the organization. Archives that are most useful tend to be private, as the public archives have no clear direction and hence suffer from lack of purpose and coherency. And finally, practice usually occurs either in an out-of-context formal setting or in a performance setting where the pressure may be high.

The combination of all these factors diminish performance and limit what can be learned. Energy and time for exploration is used up in searching for knowledge, not in trying alternative approaches. Deciding what to do is based on the most frequent or most recent experiences of the individual, not on the most relevant experiences of the community. Practice is often not followed by reflection. Solutions are often adopted because they make symptoms go away, not because a problem is understood or fixed.

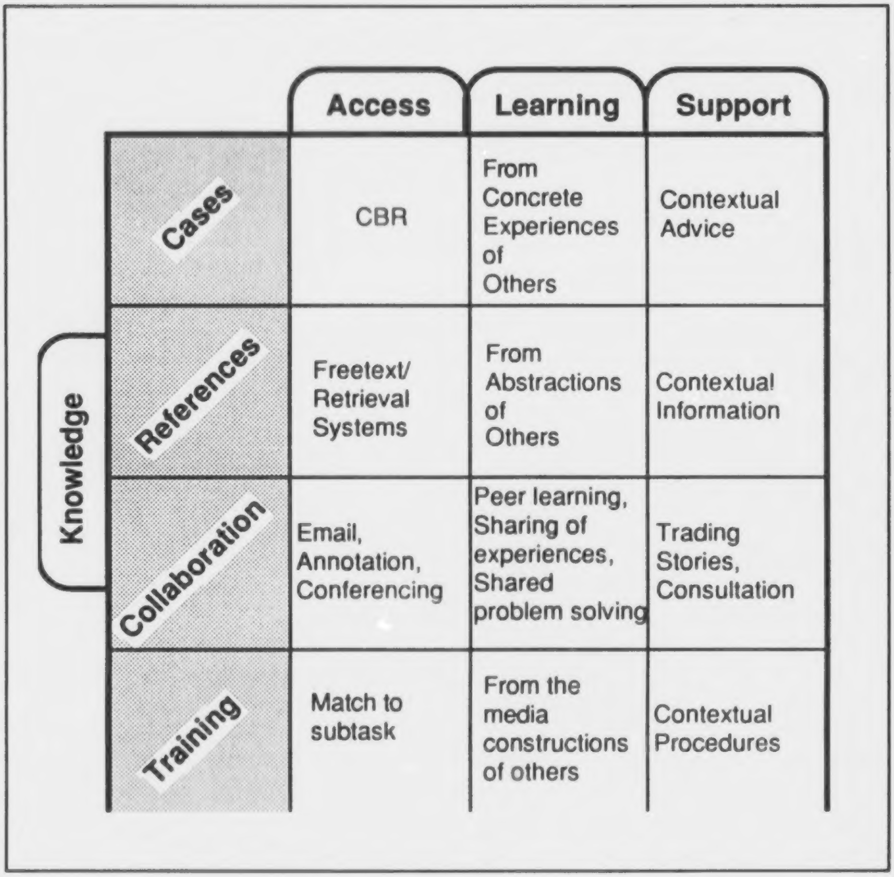
Access to organizational memory will support performance by providing relevant knowledge resources to the tasks at hand and support learning by raising confidence, encouraging exploration, and showing relationships. When knowledge workers participate in building a shared organizational memory they learn through having to articulate observations and processes and by reflecting on outcomes and relationships.

Integrated Support & Learning System

We translated our conception of organizational memory for ATC's into an integrated support and learning system (see Figure 1). The system includes resources (cases, references, collaboration,and training), access methods, and a view of how these resources are used for learning and support. The learning and support columns of Figure 1 are preliminary and only represent general abstractions for what we expect to find as we study how ATC's use this system.

For the case library implementation, we chose to use a Macintosh Allegro Common Lisp based toolkit called the "CBR Shell" from Cognitive Systems, Inc. The CBR Shell gave us a simple form based system for representing our cases. A detailed description of our use of the CBR Shell can be found in Laffey, Machiraju, Chandhok (in press).

For collaboration we will implement E-mail services with a specialized directory for matching expert collaborators. The kinds of collaboration support will include annotation and conferencing capabilities. Annotation tools will be important for supporting asynchronous collaboration while the synchronous collaboration will happen via voice (telephone) or by a conferencing tool. For example, the IS&LS will be able to recommend experts for a particular problem based on the problem description (we also envision providing "electronic secretaries" to insulate the experts from a potential flood of requests). In addition, we are exploring ways to facilitate informal collaboration through the computer



		Access	Learning	Support
Knowledge	Cases	CBR	From Concrete Experiences of Others	Contextual Advice
	References	Freetext/ Retrieval Systems	From Abstractions of Others	Contextual Information
	Collaboration	Email, Annotation, Conferencing	Peer learning, Sharing of experiences, Shared problem solving	Trading Stories, Consultation
	Training	Match to subtask	From the media constructions of others	Contextual Procedures

Figure 1. Components of an integrated support & learning system

network by setting up best-effort broadcast channels that can be filtered by the interest of the listener. Informal requests for information and help could therefore be distributed to all interested parties at a lower network cost than the guaranteed delivery of email or other media. These examples illustrate our goal of creating learning and support adhocracies to move towards a less abstracted instructional design model.

Access to archived references will be supported by full-text search across distributed databases. New retrieval technologies enable us to provide non-boolean descriptors and support an interactive retrieval process. Since the development of retrieval technology is an active area of research we have purposely not set out to develop new algorithms. Instead, we have allowed for the inclusion of many types of retrieval systems into the IS&LS where each will bring a particular benefit for the user.

Training is supported by providing on-demand lessons for tasks to be performed. Our concept is to deliver demonstrations at the appropriate grain size for the task at hand. We have begun building a library of training granules which can be accessed interactively by the ATC within the support environment. Training granules (we have also thought to call these trainemes) are produced as computer animations or digitized video. Our developers build granules by extracting frequently-used procedures from the case library or abstracting procedures from general guidelines.

Using a Case Library for Support and Learning

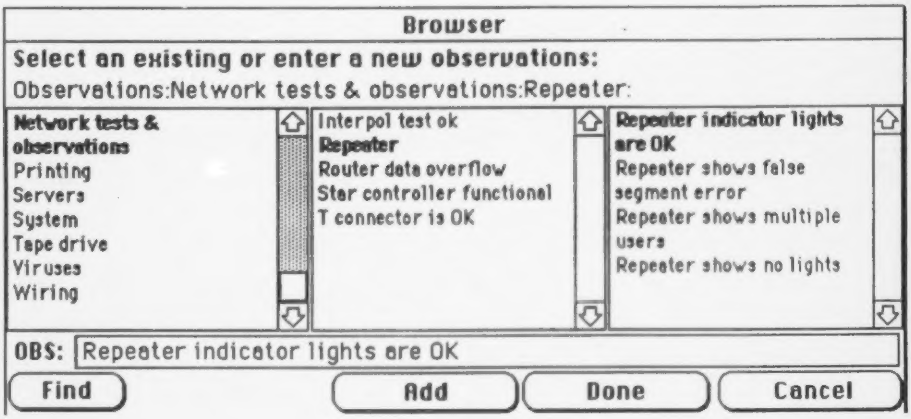
The first tests of the system have been directed at how ATC's can be supported by and learn from cases. In a laboratory environment we have experimented with interface strategies and CBR support. Volunteer ATC's have come to the lab and are presented with a Macintosh that has been "broken." We then give them as much information as a user might, and observe their approaches to solving the problem and how they utilize the NNAble in the process. Here is a typical problem solving scenario (the "user" is being supported by the ATC):

The user's message tells the ATC that after having an EtherTalk (networking) card installed they can no longer see devices on the network. Typically the ATC will call up the Chooser (a tool for finding entities on an AppleTalk network) and check for common problems and verify that the devices are really not there. ATC's with even a little experience will generally check that cables are connected. When these simple steps prove not to solve the problem, the ATC is likely to turn to NNAble. Upon entering the application the ATC is queried for a natural language (free text) description of the problem. We currently do not use this description as a filter for any searches, but future plans include evaluation of natural language-like filters. In any case, the natural language descriptions will be useful for other ATC's if the case ends up in the organizational memory. The ATC then uses a clipboard type environment to make some common and high level observations (like machine type and system version). We don't intend the ATC to make all the possible observations, but only the small set which seem relevant to the type of problem being addressed.

Now the ATC is in the main support environment where observations, procedures, notes, and a final diagnosis will be captured into a transcript of the case. A hierarchical browser facilitates entering observations in a structured fashion. Figure 2 shows an example of entering the observation "Repeater indicator lights are OK." The browser supports having the user make entries which exactly match the symbols of the system—it is our way of giving the user access to the conceptual hierarchy of the CBR Shell. As an escape, the browser also provides the ability to make new entries for new concepts.

NNAble also provides a support window for accessing procedural recommendations from the case library, people as collaborators, or reference documents. The default setting for this window is a set of common procedures for the type of problem being undertaken. These procedures are ones which occur with high frequency in the case library and are only tied to the problem type (e.g., networks, printing, applications, etc.). By checking "From Cases" the ATC activates the case retrieval system and a query is sent to the CBR knowledge base. This query is transported over the network to the Macintosh running the CBR Shell. The CBR Shell finds the most similar cases and returns the list. The NNAble interface displays the diagnosis and initial description from each case returned. By selecting a diagnosis the ATC can see the list of procedures from that case (see Figure 3) or show a complete description of the case.

In addition to the list of cases, we have augmented the case retrieval mechanism to also return a list of questions that could be used to focus the case description. In reality these questions are formulated from the split points of the decision tree in the case-based


Browser
Network tests & observations
Printing
Servers
System
Tape drive
Viruses
Wiring
Interpol test ok
Repeater
Router data overflow
Star controller functional
T connector is OK
Repeater indicator lights are OK
Repeater shows false segment error
Repeater shows multiple users
Repeater shows no lights
Find
Add
Done
Cancel

Select an existing or enter a new observations:
Observations:Network tests & observations:Repeater:

OBS: Repeater indicator lights are OK

Figure 2. Hierarchical browser for entering observations

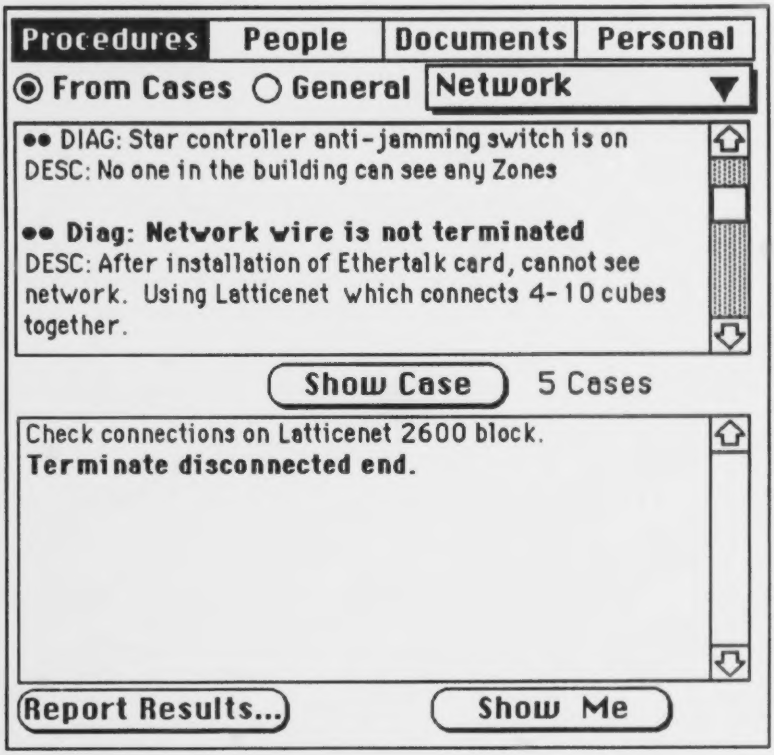
reasoning tool. The net effect, however, is to have a dynamic discrimination tree available to help the ATC solve a problem. This dynamic behavior is in stark contrast to many decision tree based systems where the decision tree is hard-wired. Since the CBR Shell re-computes the discrimination tree based on the information content of the case library, the pertinent questions adapt to meet the needs of a changing work environment.

If the ATC wants to try a procedure, but does not know how he can select the procedure and click the "show me" button. NNAble will retrieve the training granule from its library and show a "movie" demonstrating the procedure. From the "show me" selection the ATC will also have access to hints and requirements for executing the procedure.

ATC's primarily value NNAble as an intelligent assistant which raises the number of options they have in mind to try. The more novice the ATC the earlier in the troubleshooting scenario they turn to NNAble and the more they desire deeper knowledge for each option. They want training granules, explanations and glossaries for the options presented by NNAble. For novices NNAble acts as a template for troubleshooting and supports them being more systematic and able to articulate observations and relationships. On problems where NNAble has assisted in the troubleshooting, ATC's are better able to articulate a diagnosis as opposed to being satisfied with "the symptoms have gone away" type explanation of the solution. We have even seen ATC's who after having solved the problem return to NNAble to try and develop a better understanding of the solution. One of the most gratifying elements of the testing has been the variety of ways that ATC's envision using NNAble, ways which exceed our original expectations.

Issues for Performers/Learners

What is the incentive to the individual worker to augment the knowledge base? While getting support is an incentive to using the system, it is not an incentive for augmenting the



● From Cases ○ General Network ▼

•• DIAG: Star controller anti-jamming switch is on
DESC: No one in the building can see any Zones

•• Diag: Network wire is not terminated
DESC: After installation of Ethertalk card, cannot see network. Using Latticenet which connects 4-10 cubes together.

5 Cases

Check connections on Latticenet 2600 block.
Terminate disconnected end.

Figure 3. Support environment for suggested procedures

knowledge base. Building IS&LS requires understanding incentives and success measures in the workers' culture. For example, the most capable workers probably have the least incentive to use the system, but also have the most value to add to the system. How can they be encouraged to contribute? Also, once a user has solved the immediate problem, what is the incentive to completing the case with a final diagnosis for inclusion in the case library? We need to understand ways to encourage users to augment the knowledge base.

NNAble has been shown to support the troubleshooting performance of ATC's, but we know little about how it impacts competency outside the particular instance of troubleshooting. How is the individual's share of organizational memory changed through use of NNAble? Is there transfer from one problem to the next?

Issues for Organizations

Building organizational memory can improve organizational competence, but it requires investment and changes in perspective, practice and technology. IS&LS implies an integration of the individual's expertise and that of the organization, but control and decision making rest at the individual's level. What is the relationship of the individual to the organization, and on what basis will individuals increase their commitment to building a shared knowledge base. While investments in shared knowledge are in the long term interest of the organization, it is not a particularly compelling reason for the individual to relinquish a private source of control and, in some sense, power.

Issues for Technology Developments

How do we structure case libraries to build more general, larger integrated systems? The problem of mapping from a real-world problem to a detailed case is still one of great difficulty. The solution to that problem lies in natural language understanding which is still not up to the demands of our task. In addition, we have purposefully kept the case-based reasoning component simple and brute-force to reduce our dependence on the quirks and idiosyncrasies of a particular implementation. Our long term goal is to build a hybrid system to bring the strengths of various technologies to bear in a holistic fashion. From the hardware point of view, NNAble depends upon a technology platform for mobility, connectedness, intelligent applications and multimedia. Technology systems are advancing in each area, but today tradeoffs are required amongst the different functions. Again, we have opted to take advantage of off-the-shelf equipment and concentrate on the system as a whole. We expect that the hardware will get faster and more rich (in terms of multimedia) more quickly than we can develop new systems for learning and performance support.

Conclusions

In response to the changing world of work we have developed a prototype *integrated support and learning system* to augment the way people build and gain access to organizational memory. At the core of our system design is a shared organizational memory that is multifaceted and rich with multiple forms of information from reference material to multimedia presentations. In particular, we are using a case-based reasoning system to facilitate retrieval of contextually relevant information to be used directly during a problem solving effort. The case-based reasoning system has allowed us to experiment with a more dynamic system than previously possible in a standard "authoring" environment. We have also relied on network connections and multi-media to raise the information bandwidth between the user's of the system and the organizational memory.

From the early testing of NNAble we have drawn two general conclusions. First, systems like this (located in the work situation, tailored to the way knowledge workers perform, and connected via an intelligent filter to a case library of solved problems) augment the competency of the knowledge worker. Without NNAble even relatively experienced ATC's can have difficulty solving technically fairly simple problems. These are problems that if we articulated the problem to the ATC and asked him or her to solve, they would have little difficulty in coming up with a solution. But, when the problem is mired in the complexity of a situation, it is hard to recognize. Second, the ATC's find the NNAble approach attractive in that they feel in control of the system and understand the outputs of the system and how to translate system suggestions into their own practices. We attribute this second outcome to our intensive study of the ATC worker and building a system around the knowledge that is relevant to their practices.

Note

1. Apple, AppleTalk, EtherTalk, Macintosh Allegro Common Lisp and Macintosh are registered trademarks of Apple Computer, Inc.

Acknowledgements: We would like to acknowledge the ATC's of Apple Computer, Inc. Also, we would like to thank Catherine A. Abbott, Karl Anderson, Blake Roberts, Kurt Widmann and Marc Goodman of Cognitve Systems for their help. Portions of this paper have been abstracted from "NNAble: A Strategy for Learning and Performance Support." Internal document No. 102390-AT&SR-AU, Apple Computer, Inc., Cupertino, CA.

References

Brown, J.S., Collins, A. and Duguid, P. (1989). Situated Cognition and the Culture of Learning. *Educational Researcher*, January-February.

Collins, A., Brown, J.S., and Newman, S.E. (1989). Cognitive Apprenticeship: Teaching the Crafts of Reading, Writing, and Mathematics. In L.B. Resnick (Ed.), *Knowing, Learning, and Instruction: Essays in Honor of Robert Glaser.* Hillsdale, N.J.: Erlbaum.

Engelbart, D. (1962). *Augmenting Human Intellect: A conceptual framework, summary report,* SRI International. On Contract AF 49(638) 1024.

Fischer, G., Lemke, A. and McCall, R. (1990). Towards a System Architecture Supporting Contextualized Learning, *Proceedings of AAAI-1990.*

Iran-Nejad, A. (1990). Active and Dynamic Self-Regulation of Learning processes. *Review of Educational Research*, 60,4. pp 573-602.

Laffey, J., Machiraju, N. Rao, Chandhok, R. (in press). Integrated Support and Learning Systems for Augmenting Knowledge Workers: A focus on Case Based Retrieval. *Proceedings of World Congress on Expert Systems, 1991.*

Lave, J. (1988). *Cognition in Practice*. Boston MA: Cambridge.

McHale M.C. (1983). Facts & Trends: The Changing Information Environment. An Information Chartbook, Intergovernmental Bureau of Informatics, Rome.

Schon, D.A. (1987). *Educating the Reflective Practitioner*. San Francisco, CA: Jossey-Bass.

Wittrock, M. (1977). Learning as a Generative Process. In M. Wittrock (Ed.), *Learning and Instruction*. Berkeley, CA: McCutchan.

A Student-Sensitive Discourse Generator for Intelligent Tutoring Systems

James C. Lester and Bruce W. Porter
Artificial Intelligence Laboratory
Department of Computer Sciences
The University of Texas at Austin
Austin, TX, USA 78712-1188

Abstract: To communicate effectively, intelligent tutoring systems should produce clear explanations of domain.phenomena. Because the process of incorporating new knowledge into an existing set of beliefs is central to models of comprehension and learning, discourse generators for intelligent tutoring systems should be able to produce *integrative* explanations, i.e., explanations that promote the integration of new knowledge into the student's existing knowledge structures. We have developed a computational model of student-sensitive discourse generation that produces integrative explanations in the natural sciences.

An initial version of the model has been implemented in a prototype system, KNIGHT, which generates explanations in the domain of botanical anatomy, physiology, and development. KNIGHT determines the content of explanations by examining an overlay student model and finding connections between new concepts and concepts that are familiar to the student. A domain expert has analyzed KNIGHT's explanations and found them to be accurate and clear.

Introduction

Intelligent tutoring systems (ITSs) should be able to generate clear explanations of phenomena in their domains. In particular, an ITS's discourse generator should produce explanations that are appropriate for the student's current mastery of domain concepts. This student-sensitivity is important because different students vary dramatically in their knowledge of a domain and because students become increasingly familiar with a domain as they interact with the ITS.

Because the process of incorporating new knowledge into an existing set of beliefs is central to models of comprehension and learning, discourse generators for ITSs should produce *integrative* explanations. An explanation is integrative to the extent that it facilitates the incorporation of new knowledge into the student's existing knowledge structures. This process of incorporation plays a prominent role in psycholinguistic theories of language comprehension (Haviland and Clark, 1974; Wilson and Anderson, 1986) and more recently in computational models of learning (Murray and Porter, 1989). In their work on reading comprehension, Adams and Bruce (1982) set forth the requirements for an author. These requirements apply equally well to a discourse generator for an ITS.

> The task of constructing an effective linguistic message consists of 1) correctly guessing what sorts of related knowledge the intended readers already have, 2) producing expressions that will evoke appropriate subsets of that knowledge, and 3) presenting those expressions in a way that will induce readers to interrelate the evoked knowledge into a structure that most nearly captures the meaning intended. (Adams and Bruce, 1982)

Although excellent research has been conducted outside of the ITS community on issues of context-sensitivity in explanation generation (McCoy, 1989; McKeown et al. 1985; Moore and Swartout, 1989; Paris, 1988), generating explanations by searching for connections between new concepts and concepts that are already familiar to the student poses a difficult problem. To address this problem, we have designed a model of discourse generation that constructs integrative explanations. Discourse generation is typically decomposed into two subtasks: *content determination*, which is determining what to say, and *realization*, which is deciding how to say it. This paper discusses content determination procedures for selecting the content of integrative explanations. Our research also addresses issues in content organization, which is discussed elsewhere (Lester, in press). An initial version of the model has been implemented in an experimental system, KNIGHT, which generates explanations in the domain of botanical anatomy, physiology, and development.

The Architecture

Our model consists of five components: an explanation planner, a knowledge base, a user model (for ITSs, this is referred to as the student model), a query interpreter, and a linguistic realizer (Figure 1). Explanation generation begins when a query is posed, either by the student or by an ITS's pedagogical planner (Murray, 1990; Peachey and McCalla, 1986; Woolf and McDonald, 1984), which is the controlling module of an ITS.

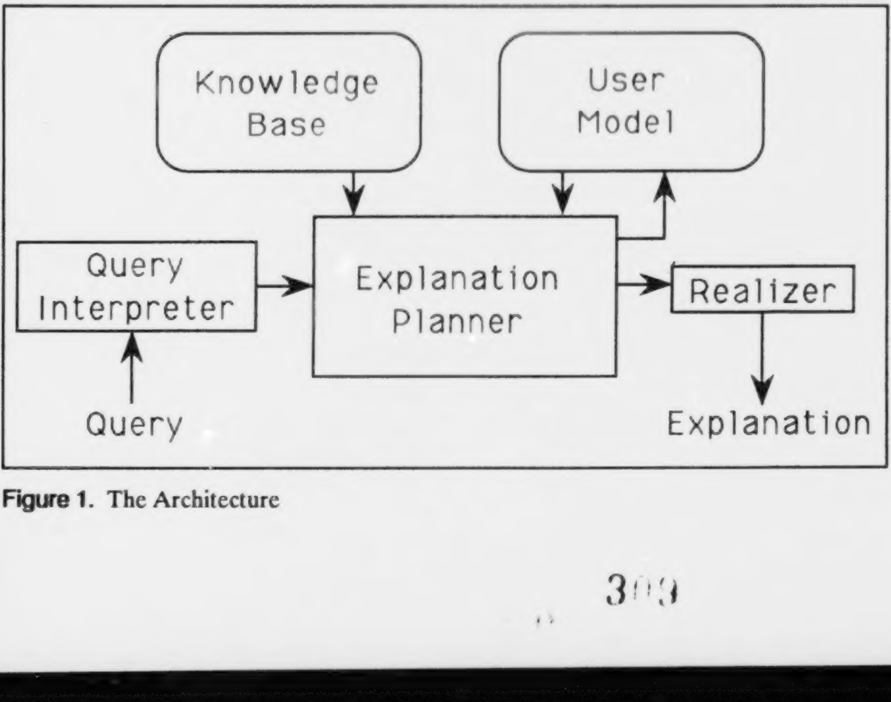


Figure 1. The Architecture

The query interpreter translates queries in natural language to canonical forms in an internal representation. The explanation planner inspects the knowledge base to obtain the raw materials for its explanations. The student model represents the student's current mastery of concepts in the domain. Our research focuses on designing a generator that exploits the student model–not on developing methods for inferring the content of the student model. However, to maintain a current record of the student's understanding of the domain, when an explanation is generated, the central concepts of that explanation are added to the student model. The system constructs an *explanation plan* by creating *discourse-elements*, which are specifications for sentences to be generated. The planner organizes the elements into paragraphs and sentences, and passes them to the realizer, which translates them into text.

KNIGHT's architecture adheres to the model presented above. Its knowledge base is the Botany Knowledge Base, a large-scale, frame-based representation that we and our colleagues have been constructing for the past four years (Porter et al. 1988). This knowledge base contains over two thousand frames about botanical objects (i.e., plant anatomy) and the processes that change them (i.e., plant physiology and development).KNIGHT's student model is an *overlay* student model (Carr and Goldstein, 1977), which uses the same representation language and domain knowledge as the Botany Knowledge Base. KNIGHT's input is provided in canonical form. Its realizer is a template-based natural language generator. The next section opens up the explanation planning box and discusses how it achieves student-sensitivity in content determination.

Integrative Content Determination

A discourse generator for ITSs should determine content in an integrative fashion. It can accomplish this by using an overlay student model and by applying procedures that find connections between concepts to be explained and concepts already present in the student model.

This notion of integrative connections can be formalized in a graph-theoretic manner. It is useful to define three networks of relevant concepts and relations for integrative content determination (Figure 2). The target *network* is the set of concepts and relations that the planner seeks to communicate to the student. The *base network* is the set of concepts and relations that model what the student already understands and are relevant in some way to the target. The *connecting network* is the set of concepts and relations that relate the target to the base.

When presented with a query, KNIGHT forms target networks by using *graph schemata* to extract a subgraph of the knowledge base. The schemata were derived from an analysis of information conveyed in science textbooks. KNIGHT instantiates the graph schema associated with the type of the query. It has graph schemata for describing objects and for explaining processes. For example, to explain a process *P*, it applies a process-description graph schema to define a graph consisting of *P* itself, the actors in *P*, the location of *P*, the subprocesses of *P*, and the subprocesses' actors. When applied to the concept of embryo sac formation in the Botany Knowledge Base, the process graph schema extracts a target network for this concept, which consists of the bottom three levels of the semantic network shown in Figure 3.

Once it has determined the target network, KNIGHT uses connection procedures to find the connecting and base networks. The procedures take "views" on a knowledge base as guidelines for determining appropriate connections. Suthers suggests that an explanation

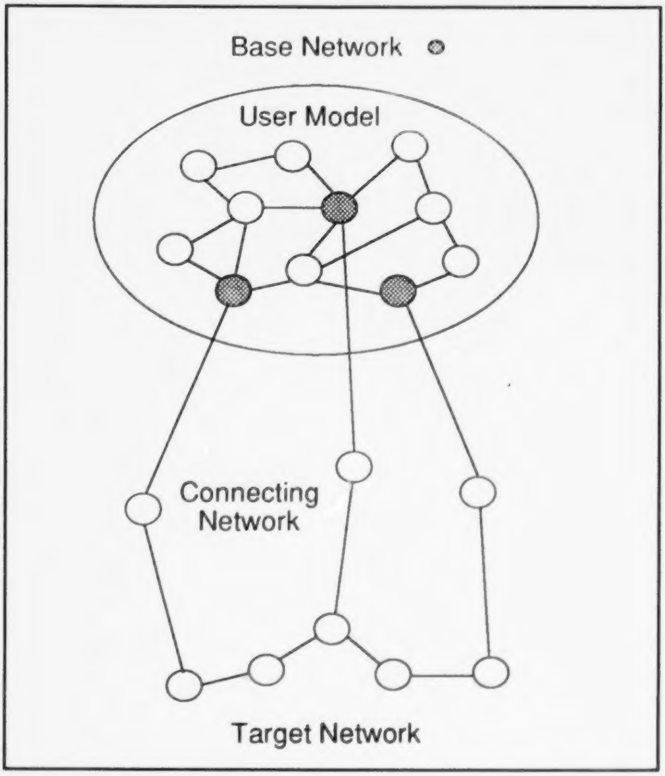


Figure 2. Integrative Content Determination

generator should access views on a knowledge base by attending to the kind of knowledge involved (Suthers, 1988). Similarly, members of the Botany Large-Scale Knowledge Base Project have developed *views* (Murray and Porter, 1989) and *view types* (Acker et al. 1991) to specify coherent networks in knowledge bases.

In particular, networks can be specified with the categorical, functional, structural, and modulatory view types. The *categorical* view type considers a concept in terms of how it fits into a concept hierarchy. The *functional* view type considers the role of an object in a process. The *structural* view type considers the structures that an object is part of as well as the parts of an object; it also considers the processes of which a process is a step, and the steps of a process. The *modulatory* view type considers how an object or process affects or is affected by another object or process. These view types focus the spreading activation search for connections from target concepts to base concepts.

Given a concept in the target network, the connection procedures use the view types to determine the relevant concepts that are familiar to the student. KNIGHT uses different connection procedures for objects and processes, which are the two basic elements of ontologies for physical domains. To provide an integrative explanation of an object O1, KNIGHT examines the knowledge base and the student model to search for the connections from *O1* to:

- the most specific concept *O2* that is known to the student and that lies in the concept hierarchy above *O1* (a categorical connection)
- the lowest object *O2* in the partonomy (the breakdown of objects into increasingly smaller parts) that is known to the student and of which *O1* is directly or indirectly a part a structural connection)
- some process P that is known to the student and in which *O1* participates (a functional connection)

Similarly, to provide an integrative explanation of a process *P1*, KNIGHT examines the knowledge base and the student model to search for the connections from P1 to

- the most specific concept *P2* that is known to the student and that lies in the concept hierarchy above *P1* (a categorical connection)
- the lowest process *P2* in the processonomy (the breakdown of processes into increasingly more detailed steps) that is known to the student and of which *P1* is directly or indirectly a step (a structural connection)
- some process *P2* that is known to the student and that immediately precedes or succeeds *P1* in time (includes both structural and modulatory connections)

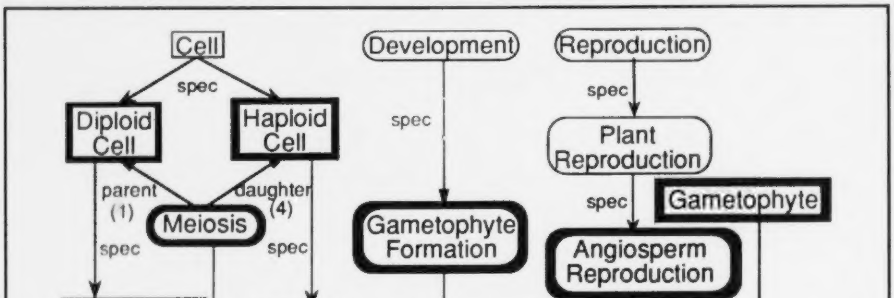

Cell
spec
Diploid Cell
Haploid Cell
parent (1)
daughter (4)
Meiosis
spec
spec
Spore Mother Cell
Spore
Development
spec
Gametophyte Formation
spec
Reproduction
spec
Plant Reproduction
spec
Gametophyte
Angiosperm Reproduction
subevent
subevent
Ovary
Pollen Grain Transfer
part
spec
occurs-before
location
Ovule
spec
spec
spec
dev-before
Embryo Sac Formation
dev-after
Embryo Sac
subevent
subevent
subevent
Megasporogenesis
Megaspore Degeneration
Embryo Sac Development
dev-after
parent (1)
daughter (4)
degen-thing (3)
dev-before (1)
Megaspore Mother Cell
Megaspore

Figure 3. A Knowledge Base Fragment

The connection procedures identify the networks of concepts and relations that are relevant to concepts in the target network and are familiar to the student. To illustrate, KNIGHT was asked, "How does an embryo sac form?" The student model representing the student posing the query is shown in Figure 3. (The concepts that are familiar to the student are indicated by the darkened outlines.)

KNIGHT first identifies the target network by applying the *process-description* graph schema. Then it searches for connections from concepts in the target network to relevant concepts that are already known to the student. For example, to find a categorical connection from *megaspore mother cell* to a relevant concept that is known by the student, KNIGHT climbs the hierarchy to locate the first generalization of this concept that is familiar to the student. The student is unfamiliar with *megaspore mother cell*, so KNIGHT climbs to the generalization of this concept, which is *spore mother cell*. The student is unfamiliar with this concept as well, so KNIGHT climbs another generalization link to *diploid cell*, which is known to the student. It then constructs a discourse-element to inform the student that a megaspore mother cell is a type of diploid cell.

KNIGHT finds similar kinds of connections between *megaspore* and *haploid cell*, between *megasporogenesis* and *meiosis*, and so forth. It creates discourse-elements that explain these connections and translates them to natural language. This results in a three-paragraph exposition that explains, in terms that are familiar to the student, the mechanisms underlying the formation of embryo sacs.

Conclusion

To maximize their educational potential, ITSs should produce explanations of domain phenomena that are appropriate for the current student. We have described a computational model for achieving this context-sensitivity in discourse generation. The generator determines the content of integrative explanations by employing our graph-theoretic formulation of the content determination problem. It uses view types to control the search for connections between concepts in the target network and concepts in the base network.

We have implemented this method in a prototype system, KNIGHT, whose explanations have been analyzed by a domain expert. He found the explanations to be accurate and clear. These favorable initial results suggest a promising path toward significantly improving the communicative capabilities of ITSs.

This research is supported by the National Science Foundation under grant IRI-8620052; the Air Force Human Resources Laboratory under grant F33615-90-C-0014; and donations from the research laboratories of Hughes and GTE.

Acknowledgements: We are grateful to those responsible for the construction of the Botany Knowledge Base. These include Art Souther, Karen Pittman, Ken Murray, Liane Acker, Brad Blumenthal, Dave Severinson, and Tom Jones. KNIGHT was implemented using Theo, a knowledge base access tool provided by Tom Mitchell. Thanks to Erik Eilerts for extending Theo's capabilities significantly and for building Kned, a knowledge base editing tool. Thanks also to Brad Blumenthal for his Lisp wizardry. Thanks to Rich Mallory, Kathy Mitchell, Ken Murray, Hwee Tou Ng, Jeff Rickel, Wing Wong, and especially to Liane Acker for helpful discussions and comments on earlier drafts of this paper.

References

Acker, L., Lester, J., Souther, A., and Porter, B. (1991). Generating Coherent Explanations to Answer Students' Questions. In H. Burns, J. Parlett, and C. Redfield (Eds.), *Intelligent Tutoring Systems: Evolutions in Design* (pp. 151-176). Hillsdale, New Jersey: Lawrence Earlbaum.

Adams, M. and Bruce, B. (1982). Background Knowledge and Reading Comprehension. In J. Langer and M. Smith-Burke (Eds.), *Reader Meets Author / Bridging the Gap: A Psycholinguistic and Sociolinguistic Perspective* (pp. 2-25). Newark, Delaware: International Reading Association.

Carr, B. and Goldstein, I. (1977). *Overlays: A Theory of Modelling for Computer Aided Instruction*. (Artificial Intelligence Laboratory Memo 406). Boston, Massachusetts: Massachusetts Institute of Technology.

Haviland, S. and Clark, H. (1974). What's New? Acquiring New Information as a Process in Comprehension. *Journal of Verbal Learning and Verbal Behavior* 13: 512-521.

Lester, J. (in press). *Generating Context-Sensitive Explanations in Interactive Knowledge-Based Systems*. (Artificial Intelligence Laboratory Tech. Rep.). Austin, Texas: Department of Computer Sciences, University of Texas at Austin.

McCoy, K. (1989) Generating Context-Sensitive Responses to Object-Related Misconceptions. *Artificial Intelligence* 41: 157-195.

McKeown, K., Wish, M., and Matthews, K. (1985). Tailoring Explanations for the User. *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 794-798.

Moore, J. and Swartout, W. (1989). A Reactive Approach to Explanation. *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1504-1510.

Murray, K. and Porter, B. (1989). Controlling Search for the Consequences of New Information During Knowledge Integration. *Proceedings of the Sixth International Workshop on Machine Learning*, 290-295.

Murray, W. (1990). A Blackboard-Based Dynamic Instructional Planner. *Proceedings of the Eight National Conference on Artificial Intelligence*, 434-441.

Paris, C. (1988). Tailoring Object Descriptions to a User's Level of Expertise. *Computational Linguistics* 14(3): 64-78.

Peachey, D. and McCalla, G. (1986). Using Planning Techniques in Intelligent Tutoring Systems. *International Journal of Man-Machine Studies* 24: 77-98.

Porter, B., Lester, J., Murray, K., Pittman, K., Souther, A., Acker, L., and Jones, T. (1988). *AI Research in the Context of a Multifunctional Knowledge Base: The Botany Knowledge Base Project*. (Artificial Intelligence Laboratory Tech. Rep. AI88-88). Austin, Texas: Department of Computer Sciences, University of Texas at Austin.

Suthers, D. (1988). Providing Multiple Views of Reasoning for Explanation. *Proceedings of the International Conference on Intelligent Tutoring Systems*, 435-442.

Wilson, P. and Anderson, R. (1986). What They Don't Know Will Hurt Them: The Role of Prior Knowledge in Comprehension. In J. Orasana (Ed.), *Reading Comprehension: From Research to Practice* (pp. 31-48). Hillsdale, New Jersey: Lawrence Earlbaum.

Woolf B. and McDonald, D. (1984). Building a Computer Tutor: Design Issues. *IEEE Computer* 17(9): 61-73.

A Framework for Improving Students' Comprehension of Word Arithmetic and Word Algebra Problems

Anne Bovenmyer Lewis and Mitchell J. Nathan
Institute of Cognitive Science
University of Colorado, Campus Box 345, Boulder, CO 80309

Abstract: We present a text-comprehension based theory of word problem-solving processes that makes a distinction between comprehending the surface structure of a problem, the situation it describes, and its underlying formal mathematical structure. Two empirical studies that applied the theory to an instructional setting in an effort to improve students' problem comprehension are reported. One study was conducted with the domain of arithmetic word problems, while the other was conducted with the domain of algebra story problems. Both involved teaching students to use external, spatial, graphic representation methods and required students, personally, to assess the correctness of their formal, equation representations. In both investigations, students who were trained performed significantly better than controls on problems like those used in training and on problems designed to assess transfer of trained skills. We conclude that teaching students to concretize the relationship between the situations described in problem texts and the mathematical equations needed to express this relationship can be a valuable addition to mathematics curricula.

Students have particular difficulty generating a set of solvable mathematical equations from problem texts (DeCorte, Verschaffel, & De Win, 1985; Dossey, Mullis, Lindquist, & Chambers, 1988; Lewis & Mayer, 1987). Therefore, many researchers have called for more explicit instruction in the representation skills needed for comprehending the mathematical structure of word problems (Hall, 1989; Lewis, 1989a; Nathan, Johl, Kintsch, & Lewis, 1989; Silver, 1987). This paper outlines a text-comprehension based theory of problem solving, then presents the results of two empirical studies, which translated these theoretical ideas to instructional design.

Solving word problems involves two major processes, problem representation—reading the problem and deriving a mathematical structure reflecting the situation described in the text—and problem solution—planning and carrying out the necessary computations to arrive at a numerical answer (Mayer, 1985). Our focus is the representation stage of problem solving, which can be characterized as a special type of text comprehension. Kintsch and Greeno (1985) proposed that when reading a word problem, students first construct a "textbase," or representation of the textual input itself. Along with this textbase, a "situation model" is formed. A situation model represents the actions of the text in everyday, qualitative terms. Finally, in a later version of the theory (Nathan et al., 1989), students must integrate and quantify the situation to create a formal "problem model," which represents the problem situation in mathematical terms, e.g., equations. It should be noted that textbase and situation model construction are similar to the problem-solving stage of "translation" described by Mayer (1985), whereas problem-model formation is parallel to Mayer's "problem-integration" stage.

Thus, students must make a correspondence between their informal understanding of a problem situation and the formal equations needed to mathematically represent that situation. Our studies test the effectiveness of training that is directed at improving students' ability to make this correspondence, by making it more explicit through use of external, graphic representation schemes. Although one study focused on arithmetic word problems and the other focused on algebra story problems, both instructional techniques require that students, themselves, compare a spatial representation with their own understanding of problem situations, and neither instructional method dealt with the problem-solution phase of the problem-solving process.

Study 1: Arithmetic Word Problems

This study was conducted to assess the impact of training students to use a paper-and-pencil diagramming technique to represent the relationship between variables in Compare arithmetic word problems. Compare problems are characterized by the presence of relation statements, sentences that express the value of one variable in terms of another variable, such as "Jeff is 10 cm shorter than Tom." Compare problems were targeted for training because relation statements have consistently proven difficult for students to comprehend (Briars & Larkin, 1984; Cummins, Kintsch, Reusser, & Weimer 1988; Kintsch & Greeno, 1985; Lewis & Mayer, 1987; Mayer, 1982; Morales, Schute, & Pellegrino, 1985; Riley, Greeno, & Heller, 1983).

Consider the following compare problem:

Problem 1

At ARCO gas sells for $1.13 per gallon.
This is 5 cent less per gallon than gas at Chevron.
How much do 5 gallons of gas cost at Chevron?

To solve this problem, the arithmetic relation between the cost of gas at ARCO and the cost of gas at Chevron must be comprehended. An incorrect selection of subtraction as the necessary operation (i.e., $1.13 - .05 = ?) is called a "reversal" error, the predominant error made on compare problems (Lewis & Mayer, 1987).

Because students have difficulty relating their equation choice to the situation described in a problem's text, a simple graphic method was developed to serve as an external, spatial representation of the relative values of problem variables. This spatial representation can be easily compared with the problem's text and then converted into a proper equation. Figure 1 outlines this diagramming method and the instruction that accompanied it.

Method

From a pool of 299 pretested college students, 96 students who made reversal errors were assigned to one of three 32-subject groups in a pretest-posttest control group design. The pretest/posttest presented 8 "target" compare problems, which included one relational statement, 4 "transfer" compare problems, which included two relational statements, and 10 "filler" problems, which did not have relational statements.

After pretesting, subjects participated in two 1/2 hour training sessions and one 1/2 hour posttesting session. Subjects in the "diagram" group were first taught to identify relation statements and then taught how to diagram the relation between variables in problems with one relation statement (target problems), as shown in Figure 1. Subjects in the "statement" group were only taught to identify relation statements and advised to be cautious with them. Subjects in the "control" group received no training but were exposed to the same problems as the other subjects. For a more detailed account of this method, see Lewis (1989b).

Results and Discussion

Comprehension of compare-problem structure was measured by the probability of reversal errors for pre- and posttest target and transfer problems. Figure 2 presents target-problem data for each of the three subject groups. A two-way ANOVA revealed a significant Training Treatment X Time of Test interaction, $\underline{F}(2,93) = 3.28$, $\underline{p} < .05$, $\underline{MS}_e = 0.5$, indicating a difference in the effectiveness of the three groups' training. A Post-hoc Newman-Keuls comparison ($\underline{p} < .05$) showed that the diagram group made significantly less posttest target-problem reversal errors than either the statement or control groups, which did not differ from each other. Furthermore, the diagram group displayed greater pretest-posttest performance gains than both the statement and control groups.

When solving posttest transfer problems, all 32 diagram-group subjects spontaneously applied the diagramming method in an iterative fashion. Figure 3 presents transfer-problem data for each of the three subject groups. Again, a two-way ANOVA revealed a significant Training Treatment X Time of Test interaction, $\underline{F}(2,93) = 4.01$, $\underline{p} < .05$, $\underline{MS}_e = 0.2$. Supplemental analyses showed that the diagram group made significantly fewer reversal errors at posttesting than at pretesting, $\underline{t}(31) = 3.40$, $\underline{p}$,.001, while no significant improvement was indicated for the statement or control groups.

Thus, subjects in the statement group, who learned to identify troublesome relational statements did not improve their comprehension of the conceptual structure of compare problems beyond the improvement of the control group. On the other hand, subjects in the diagram group, who learned to draw integrated diagrams to represent the relation between compare-problem variables, improved their comprehension more than the other groups. Use of an external, spatial diagram appears to have aided them in making the crucial connection between the situation expressed in a problem's wording and the arithmetic operation, or equation, needed for solution.

Study 2: Algebra Story Problems

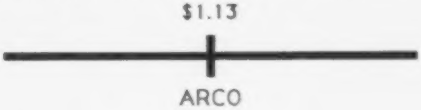
Our second study extended the results reported above to the domain of algebra story problems and utilized a computer-based tutor to enable animated, dynamic representation of problem situations. ANIMATE, written in HyperCard^{TM}, runs on Apple^{TM} Macintosh computers and currently tutors problems in the AMOUNT- PER-TIME RATE family of algebra problems (Mayer, 1981). These problems are characterized by an underlying "distance = rate x time" mathematical structure. For example:

Sample Problem

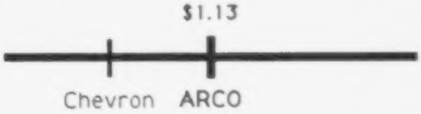
At ARCO gas sells for $1.13 per gallon. This is 5 cents less per gallon than gas at Chevron. How much do 5 gallons of gas cost at Chevron?

Diagramming Steps

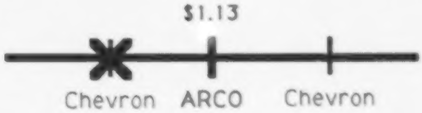
1. Draw a number line and place the variable and value from the assignment statement at the middle of the line.



2. Tentatively place the unknown variable (Chevron's gas price) on one side of the middle.



3. Compare your representation with the information in the relation statement, checking to see if your representation agrees with the meaning of the relation statement. If it does, then you can continue. If not, then try the other side.



4. Translate your representation into an arithmetic operation. If the unknown variable is to the right of the center, then the operation is an increase, such as addition or multiplication. If the unknown variable is to the left, then the operation is a decrease, such as subtraction or division.

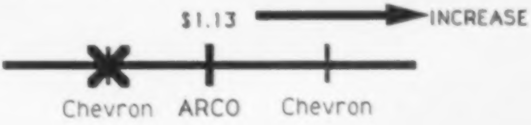


Figure 1. Diagramming procedure for the sample problem

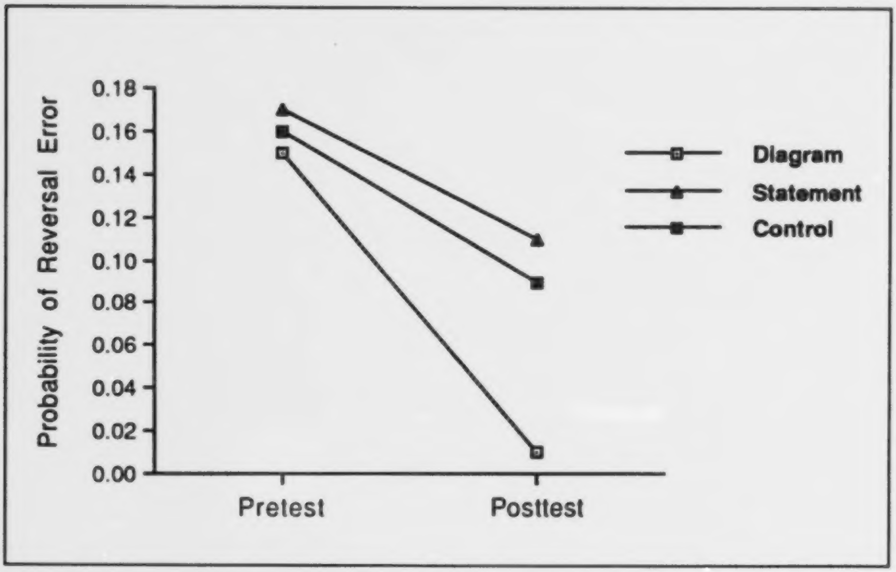


Figure 2. Probability of pretest and posttest target-problem reversal errors for the three groups in Study 1

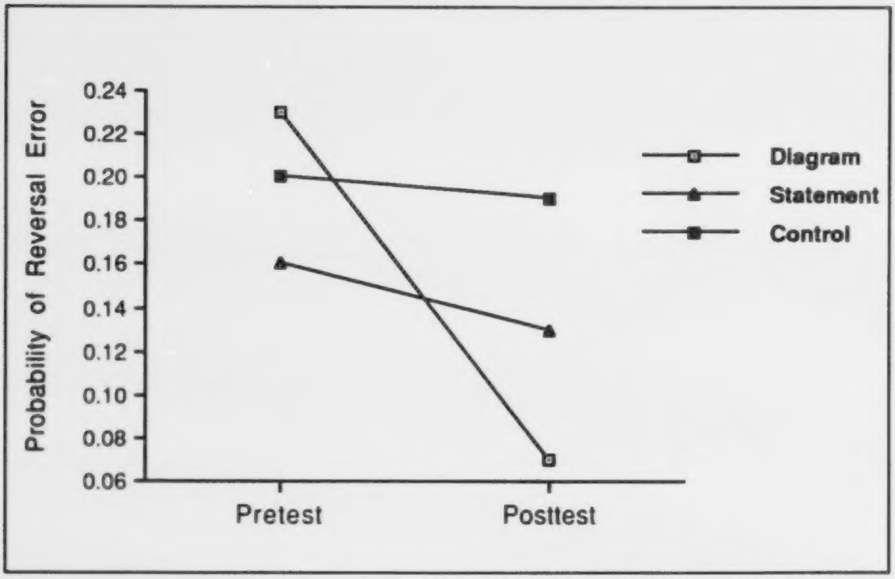


Figure 3. Probability of pretest and posttest transfer-problem reversal errors for the three groups in Study 1

Problem 2

> A huge ant is terrorizing San Francisco. It travels east toward Detroit, which is twenty four hundred miles away, at four hundred miles per hour. The Army learns of this one hour later and sends a helicopter west from Detroit at six hundred miles per hour to intercept the ant. If the ant left at 2 PM, what time will the helicopter and the ant collide (ignoring any time changes)?

In the direct-manipulation ANIMATE environment, a student selects graphics of an ant and a helicopter and specifies their starting locations and directions. The ant and helicopter are then displayed at the top of the screen as shown in Figure 4 and will serve as a representation of the situation described in the problem. Next, the student begins to build a network of equations that express the formal, mathematical structure of the problem situation. This is accomplished by choosing equations from a palate of options. The student chooses one equation as a "stopping condition"–a mathematical relation that will be true when the situation ends. An example would be when distances are equal for overtake problems, such as, when the sum of the distances travelled equals 2400 miles in Problem 2 shown above. Finally, the student directs ANIMATE to put the ant and helicopter into motion in accordance with the newly constructed formal network. In this manner, the student can see the relative motions of the ant and helicopter that the created formal representation dictates. If the characters' movements do not correspond to the student's expectations from his or her understanding of the problem situation, the student learns that the equation network is incorrect and may reorganize the network by changing variables, values, and/or operators until the expected animation is generated.

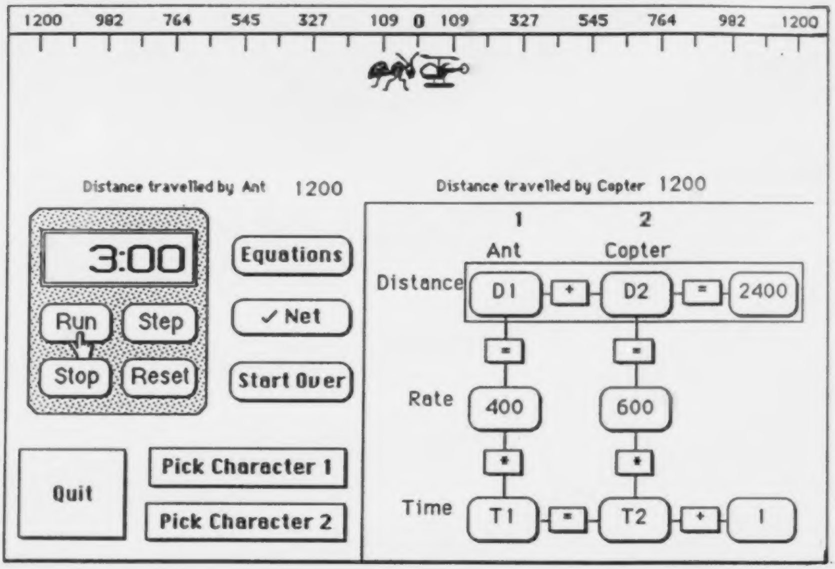

1200
982
764
545
327
109
0
109
327
545
764
982
1200
Distance travelled by Ant 1200
Distance travelled by Copter 1200
3:00
Run
Step
Stop
Reset
Equations
✓ Net
Start Over
Quit
Pick Character 1
Pick Character 2
1
2
Ant
Copter
Distance
D1
+
D2
=
2400
=
=
Rate
400
600
*
*
Time
T1
=
T2
+
1

Figure 4. The ANIMATE environment. This example shows a collision situation in accordance with expectations. Time and distance gauges show solution values of the network formalism.

Method

Fifty-six college students were randomly assigned to one of four groups in a pretest-posttest control group experimental design and participated in a two hour session that encompassed pretesting, training, and posttesting. The pretest and posttest consisted of four paper and pencil problems: two target problems, which were standard distance-rate-time problems, one near- transfer problem, which required students to debug erroneous equation representations of a problem, and one far-transfer problem, which directed students to write a short story corresponding to a set of presented equations.

Each of the four groups was exposed to a different training environment while learning to solve three training problems. The animation group used the ANIMATE tutoring environment, which included: organizing algebraic formalisms in the network graph structure, setting up the qualitative situation (e.g., selecting the characters and stopping condition), and using the equation- driven animation as feedback to assess the correctness of the equation network. The stopping-condition group had the same computer-based environment except that it lacked a running animation as feedback. The network group used the same computer environment as the stopping-condition group, except that they did not set up the qualitative situation, i.e., did not set up a picture or stopping condition. The equation group served as the ultimate control group; all of their training was done with paper and pencil and without the use of the network formalism. Results and Discussion

As was the case for the first study, what is of interest in this study is how well students comprehended the structure of pre- and posttest problems. The target problems were scored as correct if subjects generated the proper equations. Group performances are shown in Figure 5. A one-way between-subjects ANCOVA was conducted with posttest target-problem performance as the dependent measure, pretest score at the covariate, and treatment group as the between-subjects variable. A significant difference among the performances of the four groups was confirmed, $\underline{F}(1, 55) = 11.56$, $\underline{p} < .05$, $\underline{MS}_e = 2.3$. A post-hoc Newman-Keuls comparison ($\underline{p} < .05$) revealed that the animation group performed significantly better than the other three groups, whereas the other groups did not differ from each other. Subjects exposed to the ANIMATE environment had improved ability to make a correspondence between formal mathematical equations and their understanding of problem situations.

Figure 6 displays subjects' performance on the near-transfer, debugging task. Although improvement was greatest for the animation group, differences among groups were not statistically significant. Animation-group subjects, however, were more likely to repair a buggy equation if they detected it, $\underline{F}(1, 55) = 5.0$, $\underline{p} < .05$. Finally, Figure 7 presents far-transfer, story writing, performance on pre- and posttest. A one-way, between-subjects ANCOVA with posttest performance as the dependent variable, pretest score as the covariate, and treatment group as the between-subjects variable indicated a significant main effect of treatment, $\underline{F}(1, 55) = 11.8$, $\underline{p} < .002$, $\underline{MS}_e = 2.3$. A post-hoc comparison revealed that the animation group performed higher than the equation and network groups but not higher than the stopping-condition group (Newman-Keuls, $\underline{p} < .05$). Apparently, the greater exposure to situation-level processing provided by the animation and stopping-condition tutors helped subjects in this novel task by providing the graphic and highlighting the role of equations as stopping conditions in problem situations.

Thus, working in an environment that encouraged situation- based reasoning promoted better understanding of the relation between informal situation models and formal equations, even outside of the learning environment, as indicated by posttesting which did

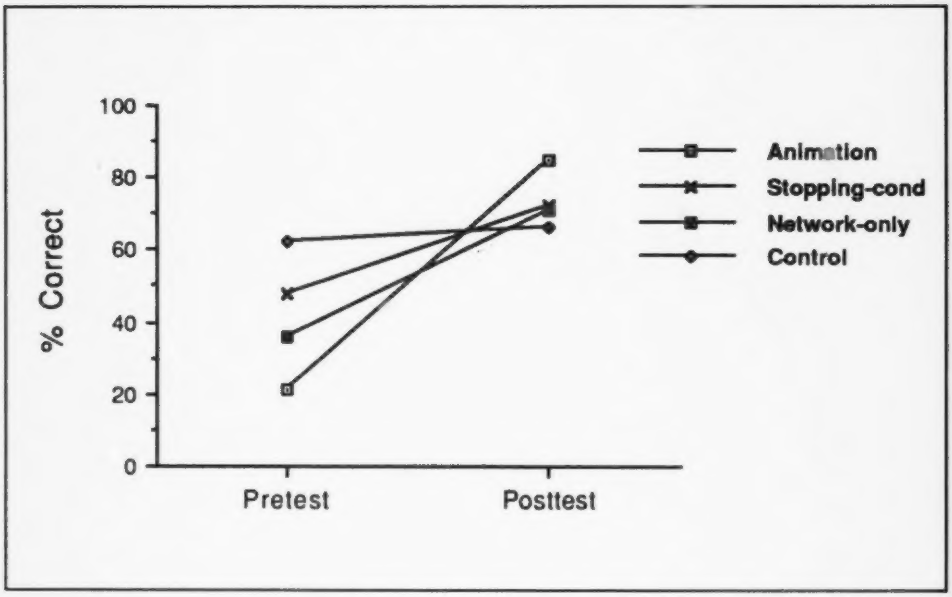


Figure 5. Percent correct on pretest and posttest target-problems for the four groups in Study 2

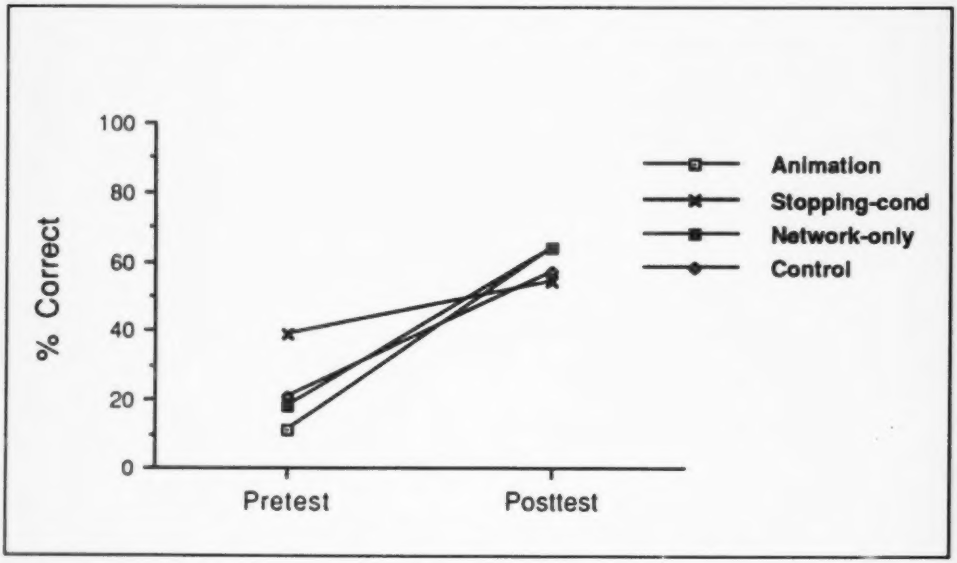


Figure 6. Percentage correct on pretest and posttest near-transfer problems for the four groups in Study 2

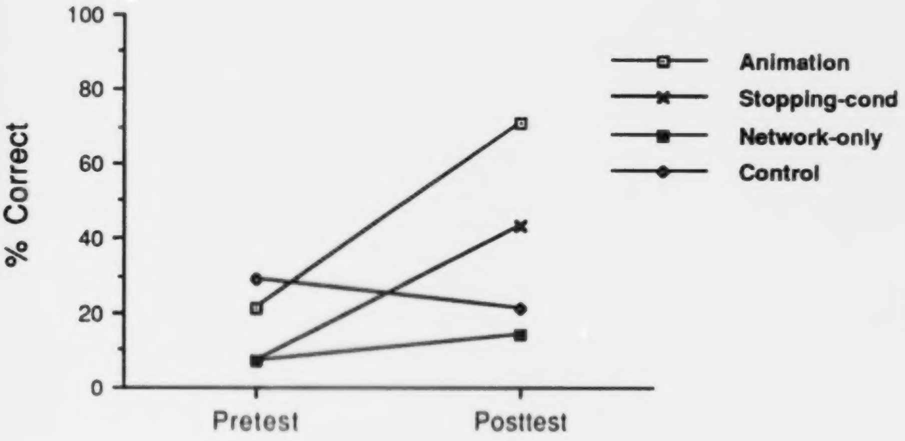


Figure 7. Percent correct on pretest and posttest far-transfer problems for the four groups in Study 2

not involve tutor use. We suggest that some important changes in conceptual understanding occurred for subjects in the situation-based learning environment that did not occur for subjects in the other training conditions.

Conclusion

The benefits of representation training are evident for both arithmetic word problems and the more complex algebra story problems. Within each domain, utilizing an external, graphic representation scheme aided students in correctly mapping formal equations to informal textual descriptions. Other researchers (e.g., Shalin & Bee, 1985; Fuson & Willis, 1989) have reported on the benefits of diagram use as an aid to problem comprehension, and the future of such endeavors appears to be bright. Of particular interest in the two representation aids reported herein is that they both involved a spatial layout of problem information and both gave students the responsibility for assessing the correctness of the correspondence between the problem situation and the formal equation(s). Furthermore, the addition of animation is a promising direction in representation-skills training (c.f., Rieber, 1990).

Acknowledgements: This work was supported in part by Grant MH15872 from the National Institute of Mental Health to Walter Kintsch and in part by a National Science Foundation graduate fellowship research grant to Anne Bovenmyer Lewis. We would also like to thank Paul Johl and Emilie Young for their contributions in developing the ANIMATE system.

References

Briars, D. J., & Larkin, J. H. (1984). An integrated model of skill in solving elementary word problems. *Cognition and Instruction*, 1, 245-296.

Cummins, D. D., Kintsch, W., Reusser, K., & Weimer, R. (1988). The role of understanding in solving word problems. *Cognitive Psychology*, 20, 405-438.

DeCorte, E., Verschaffel, L., & De Win, L. (1985). Influence of rewording verbal problems on children's problem representations and solutions. *Journal of Educational Psychology*, 77, 460-470.

Dossey, J. A., Mullis, I. V. S., Lindquist, M. M., & Chambers, D. L. (1988). *The mathematics report card: Are we measuring up?* Princeton, NJ: Educational Testing Service.

Fuson, K. C., & Willis, G. B. (1989). Second graders' use of schematic drawings in solving addition and subtraction word problems. *Journal of Educational Psychology*, 81, 514-520.

Hall, R. (1989). Qualitative diagrams: Supporting the construction of algebraic representations in applied problem solving. In D. Bierman, J. Brueker, and J. Sandberg (Eds.), *Proceedings of the 4th International Conference on AI and Education* (pp. 116-122). Amsterdam: IOS.

Kintsch W., & Greeno, J. G. (1985). Understanding and solving word arithmetic problems. *Psychological Review*, 92, 109-129.

Lewis, A. B. (1989a). Training students to represent arithmetic word problems. *Journal of Educational Psychology*, 81, 521-531.

Lewis, A. B. (1989b). *Enhancement of arithmetic word problem-solving skill through representation training*. Unpublished doctoral dissertation, University of California, Santa Barbara.

Lewis, A. B., & Mayer, R. E. (1987). Students' miscomprehension of relational statements in arithmetic word problems. *Journal of Educational Psychology*, 79, 363-371.

Mayer, R. E. (1981). Frequency norms and structural analysis of algebra story problems into families, categories, and templates. *Instructional Science*, 10, 135-175.

Mayer, R. E. (1982). Memory for algebra story problems. *Journal of Educational Psychology*, 74, 199-216.

Mayer, R. E. (1985). Mathematical ability. In R. J. Sternberg (Ed.), *Human abilities: An information processing approach* (pp. 127-150). New York, NY: Freeman.

Morales, R. V., Shute, V. J., & Pellegrino, J. W. (1985). Developmental differences in understanding and solving simple word problems. *Cognition and Instruction*, 2, 41-57.

Nathan, M. J., Johl, P., Kintsch, W., & Lewis, C. (1989). An unintelligent tutoring system for solving word algebra problems. In D. Bierman, J. Brueker, and J. Sandberg (Eds.), *Proceedings of the 4th International Conference on AI and Education* (pp. 169-176). Amsterdam: IOS.

Riley, M. S., Greeno, J. G., & Heller, J. I. (1983). Development of children's problem-solving ability. In H. P. Ginsberg (Ed.), *The development of mathematical thinking* (pp. 153-196). New York, NY: Academic Press.

Rieber, L. P. (1990). Using computer animated graphics in science instruction with children. *Journal of Educational Psychology*, 82, 135-140.

Shalin, V., & Bee, N. V. (1985). Analysis of the semantic structure of a domain of word problems. (Technical Report No. APS-20). Pittsburgh: University of Pittsburgh Learning Research and Development Center.

Silver, E. A. (1987). Foundations of cognitive theory and research for mathematics problem-solving instruction. In A. H. Schoenfeld (Ed.), *Cognitive science and mathematics education* (pp. 33-60). Hillsdale, NJ: Erlbaum.

Overview of Object-Oriented Microworlds for Learning Mathematics through Inquiry

David McArthur and Matthew Lewis
The RAND Corporation
1700 Main St., Santa Monica, CA 90401-2138

Abstract: This paper discusses *computer microworlds* that have been designed to support the learning of specific mathematical skills involved in discovering, confirming, explaining, and communicating patterns of relationships among variables in mathematical objects. It begins by discussing the software tools themselves and how they promote learning through guided inquiry. After discussing the software kernel in general, we report the results of fielding our first microworld–*Polygons*–with high school students. In *Polygons*, students learn through *guided inquiry* about the relationships among polygon properties, including number of sides, interior angle, exterior angle, apothem angle, radius, length of side, radius, apothem, perimeter, and area. We report some interesting mathematical conjectures young students were able to discover, test, refine, and explain. In addition, because students' skills at learning through inquiry appear limited in several respects, we discuss some of the problems or barriers to learning through discovery that we noticed, and we also describe how software is designed to overcome some of these barriers.

Introduction

The past years have heard many calls to reform mathematics education (National Commission on Excellence in Education, 1983; National Council of Teachers of Mathematics, 1989; Resnick, 1987; Thompson, 1988). There is widespread agreement that many traditional skills–for example, mechanical symbol manipulation abilities–are not as important as once envisioned, in part because such manipulation is not representative of most mathematical reasoning, and because this kind of mathematics can be done by computer. Mathematics educators have suggested different kinds of skills to take the place of traditional mathematics content in the classroom. Many of the proposals have suggested changes not only to the content of the mathematics curriculum but also to how mathematics is taught. In particular, there is a growing consensus that students should learn through inquiry and through the construction of their own mathematics (National Council of Teachers of Mathematics, 1989; Papert, 1980). However, these suggestions imply substantial unresolved problems for the teaching of mathematics. What specific skills shall we focus on, what costs does learning through inquiry impose on students, what benefits offset those costs, and what tools can we provide to help students learn effectively, and to help teachers teach efficiently, through inquiry? In this paper we discuss *computer microworlds* that have been designed to support and improve the learning of specific mathematical skills involved in discovering, confirming, explaining, and communicating patterns of relationships among variables in mathematical objects.

Overview of the Microworlds

There is no rigorous analysis of mathematical discovery or inquiry comparable to more "well-defined" skills, such as, factoring quadratics or performing integration. However, following Lakatos (1976), Polya (1962), and Steen (1988), we see discovery as involving the following activities:[1]

- Gathering observations that bear on an issue or topic (an issue is not a conjecture, which must either be true or false, but more of an open question).
- Generating conjectures or hypotheses.
- Confirming a conjecture.
- Disconfirming or refuting a conjecture.
- Refining a conjecture.
- Explaining or proving a conjecture (decomposing a conjecture into "smaller" conjectures that are already regarded as proved).

Our mathematical microworlds do not support all of these facets of inquiry, nor are selected activities supported in a uniform fashion. In particular, although we have developed intelligent tutoring systems (ITS) for algebra (McArthur, Lewis, Ormseth, Robyn, Stasz, and Voreck, 1989; McArthur, Stasz, Hotta, Peter, and Burdorf, 1988), the microworlds do not actively tutor students about inquiry skills. Rather, students learn various aspects of mathematical inquiry by engaging in them, using various microworld tools and activities designed to support and encourage them. However, our algebra ITS software does fulfill several important roles at this time; for example, it supplies the equation manipulation necessary for constraint propagation.

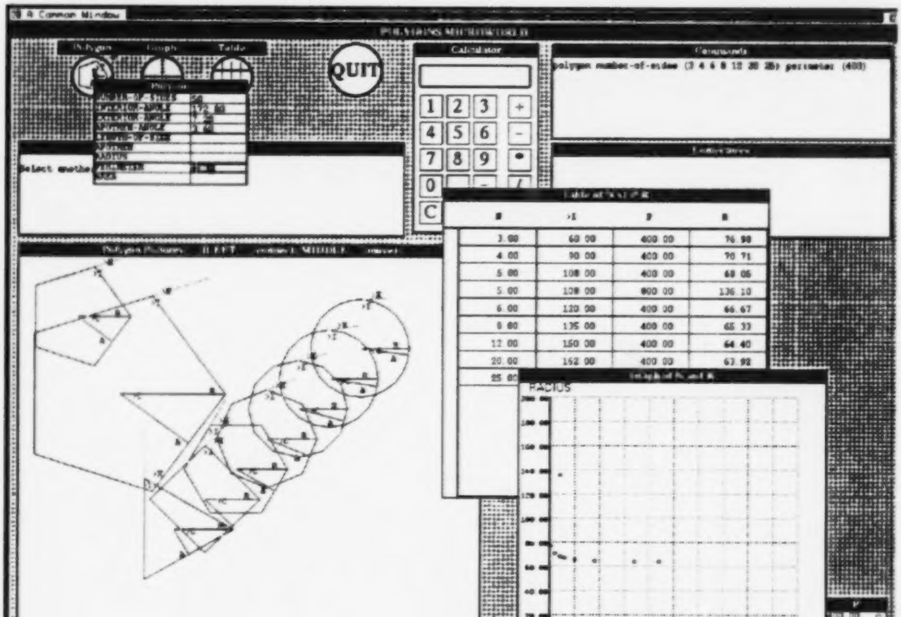
Our microworlds all share a common structure, permitting us to develop different worlds relatively easily. Central to each world is the definition of one or more types of objects. Each object possesses several different properties that are the focus of student learning. The properties are subject to several important constraints: they must have numerical values, and it must be possible to define equations that govern the relationships among properties. To define a specific microworld, we simply need to stipulate the objects, their properties, equations that relate the properties, and routines for drawing objects. These hook seamlessly into the "kernel" microworld software.

The Polygons Microworld

The examples we discuss in this paper come from students using microworlds in a "lab" setting. Typically, one or two students at a time use the software, sharing inquiries. Observers are present to chronicle events, and one is designated to interact with the students (the participant observer). Initially, the observer will act as a mentor, modeling use of the software and demonstrating how to conduct inquiries.

When a student begins working with a specific microworld, he or she sees various tools that help create instances of the objects, generate and gather data about objects, see patterns of relationships in their data, and record and test their conjectures. Figure 1 shows a version of our *Polygons* microworld. The *Polygons* properties include number of sides, interior angle, exterior angle, apothem angle, radius, length of side, apothem, perimeter, and area. As in all microworlds, students usually begin with an issue or goal. Even in a world as simple as *Polygons*, many issues can be examined, from very simple (e.g., how does interior angle vary with the number of sides?), to rather subtle (e.g., if area is held constant, what happens to apothem as the number of sides increases?); and from quantitative (e.g., find an equation that relates exterior angle to number of sides) to qualitative (e.g., how many properties need to be given to determine all others?).

Students have complete freedom to choose any issue they wish. In practice, they are usually reluctant to make such high-level decisions, and have few skills for following issues systematically. Consequently, we often use a "seeding" technique, where observers act as mentors and suggest issues, at least at the beginning of a session. (See McArthur and Lewis (1991) for a more detailed discussion of how issues are negotiated). In 1, the student's goal is to characterize the relationship between N (the number of sides of a polygon) and R (radius), holding P (perimeter) constant. Will R increase or decrease, and how can this change be characterized?



```
polygon number-of-sides (3 4 6 8 12 20 25) perimeter (400)
```

N	EI	P	R
3.00	60.00	400.00	76.98
4.00	90.00	400.00	70.71
5.00	108.00	400.00	68.06
5.00	108.00	800.00	136.10
6.00	120.00	400.00	66.67
8.00	135.00	400.00	65.33
12.00	150.00	400.00	64.40
20.00	162.00	400.00	63.92
25.00			

Figure 1. Investigation of N and R

Generally, students begin considering an issue by gathering data. Of course, for relatively simple issues, advanced students may be able to develop conjectures–even proofs–without empirical guesswork. However, for many issues, students will often have little mathematical intuition with which to generate conjectures. Data can often stimulate conjectures, and even students able to generate conjectures usually need the confirmation that data can provide. Among others, Lakatos (1976), Putnam (1975), and Steen (1988) argue that mathematicians, not just students, use "quasi-empirical" methods.

The simplest way to gather data in *Polygons* is to generate instances interactively. The student uses the mouse to click on the "Polygon" button (upper left), and a template giving all the properties of the polygon is displayed (see Figure 1). Initially, no properties have values; they are typed in by the student. Polygons have two degrees of freedom. When one scale property (radius, length of side, apothem perimeter, and area) and one angle property (number of sides, interior angle, exterior angle, apothem angle) are given, all the others can be determined. Thus, in Figure 1, after the student has selected a value for P (400) and N (50), the remaining values are computed and displayed by the system. When the student has created a specific polygon, it appears in the pictures window at the lower left.

To investigate an issue efficiently, the student needs experimentation skills. In this case, the student needs (at least intuitively) to understand the ideas of selecting a dependent variable (R), of choosing an independent variable (N and P), and of holding one independent variable (P) constant while systematically altering the other (N). Using only pencil and paper tools to explore microworlds, students would probably be limited to varying such easy-to-control properties as N or the length of a side and measuring the values of all others (although some of these, like area, might prove quite difficult to measure accurately). However, because the microworlds provide a general constraint propagation algorithm, there are no restrictions on which properties the students can choose to vary, and measurements are accurate and instantaneous. The students therefore may play rapidly any kind of "what if" game they can imagine. This freedom gives students considerable power in fashioning inquiries, but it also demands experimentation skills to use the power effectively.

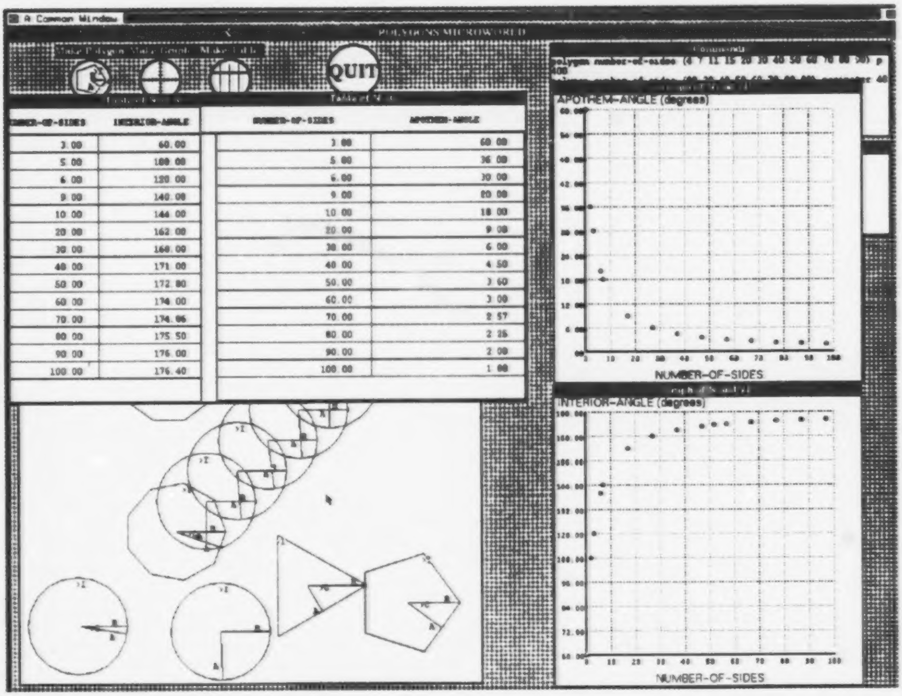
Gathered data can be represented using several microworld tools. For example, in Figure 1, the student has represented the created polygons in both a table and graph. Tables can record any number of polygon properties. Graphs only encode the relationship between pairs of properties. In general they give a selective, visual perspective on the data and are useful in describing relationships when the student has determined the specific properties of interest. On the other hand, tables can provide a broader but less vivid view of data and relationships. They may be more appropriate when students have only a vague idea of the relationships they are trying to characterize. Our limited experience suggests that students are often technically competent using tables and graphs and can interpret their meanings readily. However, the students' understanding of the uses of tables and graphs appears poorly developed. Perhaps because graphs and tables are taught in classrooms as isolated artifacts, students are often unaware of their appropriate roles as *tools* in specific inquiry activities.

Our hope is that the investigation of one issue will trigger interest in new ones that students can follow *opportunistically*. For example, in this session, the student's inquiry led to the conjecture that R decreases as N increases, and suggested to the student a new idea: Since R decreases (because the "points" of polygons are being "rounded off" as the polygons approach a circle), the apothem (A) should increase to the same asymptote (A should be equal to R as N increases to infinity). Testing this conjecture is straightforward. The student does not need to gather new data; rather, the existing polygons simply need be viewed from a different perspective. The student creates this view by generating a new graph for the properties N and A and then "connecting" the graph of N versus R to the graph of N versus A. The new graph would then show that the student's conjecture about N and A is confirmed.

This example demonstrates a general feature of our microworlds. Graphs and tables are object-oriented in the sense that they represent polygons (in this microworld), not arbitrary data values. They simply present objects from different perspectives. Thus, when we connect one data representation to another we are effectively putting the objects contained in one into the new filter or perspective represented by the other one. In general, any pair of data representations can be connected. When connected, objects in one become shared by the other, and as new objects are added to one representation, they also appear in the other. Representations can be disconnected as desired.

Quantitative Inquiries in Polygons

In the exploration discussed above, the student's conjectures were qualitative, focusing on whether one variable increases as a direct function of another, and on the shape of that relationship. Depending on the issue and on the student's skills, students may generate more quantitative conjectures. For example, 2 shows the culmination of a rather long inquiry in which the student quickly established an equation relating N and the apothem angle (>A): $>A = 180/N$.[2] To establish this rule, the student successfully used the strategy of multiplying (>A*N is a constant) and attempted to apply this or a similar strategy to describe the relationship between N and >I. This activity was to some extent motivated by the patterns in the graphs of N versus >I and N versus >A. The student had noticed the patterns were the "reverse" of one another (see the mirror-image graphs in Figure 2). Hence the student was trying to find a way of "reversing" the operation in $>A = 180/N$. Attempts to generate a constant by multiplication (N*>I) or division (N/>I) were not successful. However, when the student tried, >I/>A he found it yielded N-2, leading to the equation $>I/>A + 2 = N$.



MBER-OF-SIDES	INTERIOR-ANGLE
3.00	60.00
5.00	108.00
6.00	120.00
9.00	140.00
10.00	144.00
20.00	162.00
30.00	168.00
40.00	171.00
50.00	172.80
60.00	174.00
70.00	174.86
80.00	175.50
90.00	176.00
100.00	176.40

NUMBER-OF-SIDES	APOTHEM-ANGLE
3.00	60.00
5.00	36.00
6.00	30.00
9.00	20.00
10.00	18.00
20.00	9.00
30.00	6.00
40.00	4.50
50.00	3.60
60.00	3.00
70.00	2.57
80.00	2.25
90.00	2.00
100.00	1.80

Figure 2. Investigation of >I, >A, and N

There are several points to make about this inquiry. First, although it centered on an equational expression of a relationship, the equation was discovered by the student using some relatively sophisticated techniques; it was not given to the student. Second, notice how one conjecture built upon previously established ones. The student began with the relatively simple issue of characterizing the relationship between N and >A (this was preplanned for the student) and spontaneously used the understanding acquired there in tackling the more complex issue. The skill of building one explanation or conjecture on simpler ones–the *incremental* nature of mathematical inquiry–is common in authentic mathematical practice, although rarely treated in classrooms. Third, although in this inquiry the student formulated an equation, we placed few constraints on the structure of the equation discovered. In fact, the equation the student produced came as a novelty to the researchers observing the session. The "standard" equation is $>I = 180 - 360/N$. Making sense of the novel equation became the topic of an ensuing inquiry that included the student and observers. Fourth, we noted that the student did not spontaneously attempt to make sense of his equations in some respects. In particular, although he checked their validity empirically, the student did not naturally engage in more deductive reasoning that would explain why the equations must be true. Thus, for example, the student could not appreciate why the standard equation, $>I = 180 - 360/N$ might be preferred to forms like $>I/>A+2 = N$. The former admits of intuitively satisfying explanations along the lines that >I approaches a straight line (180) as N approaches infinity, and the difference between >I and 180 is the exterior angle, 360/N.

Finally, we wish to emphasize that, as in the first example, constructing equations is not the sole goal of inquiry, nor even the main one. Our aim is that students acquire skills in inquiry or discovery and skills in describing relationships among variables. Equations are one way to describe relationships, and they have particular strengths and weaknesses. The main strengths of equations are abstractness and power in concisely encoding a great deal of information about relationships. But this is their weakness as well. Their terseness hides many important implicit or "emergent" features of relationships that are more apparent in other representations of relationships, as we saw in the first example, where the student developed a more qualitative characterization using graphs and tables. We encourage students to generate relatively discursive, often qualitative, descriptions of relationships and to view equations as just one aspect of a complete description. The acts of establishing, confirming, explaining, and communicating the relationships are primary. Equations, as well as other representations like graphs, are useful tools in these activities.

Conclusions

Students have learned effectively using the *Polygons* microworld. They have been able to acquire various mathematical concepts, some that rarely arise in introductory high school classes and some that are usually taught more didactically. In addition, they show some ability to learn through self-directed construction of knowledge and to learn inquiry skills that would improve their skills. We have noted some apparently pervasive problems in students' inquiry skills; for example, a heavy reliance on "hacklerly reasoning" or haphazard guess-and-test. We are currently designing tools to address these limitations. (See McArthur and Lewis (1991) for a more detailed discussion of problems with inquiry learning). In spite of these difficulties we are gratified by students' persistence. The students we have piloted extensively have all learned to use the software rapidly, and, once comfortable, have happily used the system for one or two hours at a time. Rarely is there a dull moment in the sessions. The students are usually enthusiastically generating data, offering conjectures, and communicating about ideas with other students or observers.

Notes

1. We will use discovery or inquiry interchangeably in this paper.
2. On screen images in the figures, the apothem angle is referred to using the symbol "> C". This has recently been changed to "> A", and we use the latter term throughout the paper.

Acknowledgements: This research is being funded by the National Science Foundation (Applications of Advanced Technologies Program) under grant MDR-8751515. Views or conclusions expressed herein do not necessarily represent the policies or opinions of the sponsor.

References

Lakatos, I. (1976) *Proofs and Refutations*. New York: Cambridge University Press.

McArthur, D. and Lewis, M. (1991). Overview of object-oriented microworlds for learning mathematics through inquiry. N-3242-NSF, RAND Corporation, Santa Monica, CA.

McArthur, D., Lewis, M., Ormseth, T., Robyn, A., Stasz, C, and Voreck, D. (1989). Algebraic thinking tools: Supports for Modeling Situations and Solving Problems in Kids' Worlds *Technology and Learning, 3*(2).

McArthur, D., Stasz, C., Hotta, J., Peter, O., and Burdorf, C. (1988). Skill-oriented task sequencing in an intelligent tutor for basic algebra. *Instructional Science, 17*, 281-307.

National Commission on Excellence in Education (1983). *A Nation at Risk: The Imperative for Educational Reform*. Washington D.C.: U.S. Government Printing Office.

National Council of Teachers of Mathematics. (1989). *Curriculum and Evaluation Standards for School Mathematics*. Reston, VA: NCTM.

Papert, S. (1980). *Mindstorms: Children, Computers and Powerful Ideas*. New York: Basic Books.

Putnam, H. (1975). *Mathematics, Matter, and Method*. New York: Cambridge University Press.

Polya, G. (1962). *Mathematical Discovery*. New York: John Wiley and Sons.

Resnick, L. B. (1987). *Education and Learning to Think*. Washington D. C.: National Academy Press.

Steen, L. A. (1988). The science of patterns. *Science*, 240, 611-616.

Thompson, C.S., and Rathmell, E.C. (1988). NCTM's standard for school mathematics, K-12. *Mathematics Teacher*, 81 (5), 348-351.

Adapting Instruction to the Cognitive Needs of Learners in an Intelligent Tutoring System

Permanand Mohan and Jim E. Greer
ARIES Laboratory
Department of Computational Science
University of Saskatchewan, Saskatoon, Canada, S7N 0W0

Marlene L. Jones
Advanced Computing & Engineering
Alberta Research Council, Calgary, Canada T2E 7H7

Abstract: Cognitive theories of instruction have revealed that learning is an active constructive process that occurs within the learner, and which can be influenced by the learner. This paper describes a system IDP, which is built on an explicit cognitive model of instruction, worked into the architecture of an intelligent tutoring system (ITS). This model centres around a theory of "learning functions," psychological processes which must be engaged within the learner. By focussing on the cognitive processing of the learner, IDP is able to adapt instruction to the cognitive needs of individual learners.

While IDP is not a complete system, research to date indicates that it is a fruitful and desirable goal to explicitly operationalize a cognitive model of instruction in an ITS. IDP takes full advantage of the high degree of individualization offered by an ITS and provides instruction that is educationally sound. More importantly, the instruction provided is suitable for the cognitive processing needs of individual learners.

Introduction

An intelligent tutoring system (ITS) should be knowledgeable of the interactive process of teaching and learning, and should be able to tailor instruction to the learning needs of individual students. This can best be accomplished by incorporating principled theories of instruction (Jones, 1988). Few attempts have been made to design ITSs based upon instructional theories other than the traditional behaviourist paradigm of teaching and learning. In behaviourism, students are viewed as passively recording stimuli presented by the teacher (Weinstein and Mayer, 1986). Cognitive theories view learning as an active, constructive process occurring within the learner, influenced by the learner, and with success depending jointly on what is presented by the teacher, and on how the learner processes that information (Weinstein and Mayer, 1986; Shuell, 1991).

Empirical research in education has shown that significant improvements in learning occur when instruction is geared towards the cognitive processing of individual learners. For this reason, we have designed a system IDP (instructional delivery planner), which provides instruction to facilitate the cognitive processing of individual learners. Since we attempt to explicitly operationalize a cognitive model of instruction in an ITS, IDP is significantly different from systems like the LISP Tutor (Anderson et al., 1990), in which a cognitive theory is implicitly embedded in the architecture of the system.

Cognitive Model of Instruction

The cognitive model of instruction used in IDP is centred around Shuell's (1991) theory of *learning functions*. Learning functions or *cognitive operations* (Winne, 1991) are various psychological processes that must occur within the learner before learning can take place. Shuell (1991) identifies fourteen learning functions involved in the learning process. These are: *expectations*, *motivation*, *prior knowledge activation*, *attention*, *encoding*, *comparison*, *hypothesis generation*, *repetition*, *feedback*, *evaluation*, *monitoring*, *combination*, *integration*, and *synthesis*.

Learning functions can be engaged in a variety of equally effective and appropriate ways and there are two kinds of activities which can be used to engage them. These are: 1) teaching strategies, such as the teacher presenting certain material at a certain time in a certain way, and 2) learning strategies, such as the learner actively organizing or elaborating or predicting about the presented material (Weinstein and Mayer, 1986; Shuell, 1991). In IDP, knowledge of teaching strategies has been derived from various research on teaching, including Leinhardt and Greeno (1986). Knowledge of learning strategies was extracted from sources such as Weinstein et al. (1988). Two types of learning strategies are represented: *embedded* strategies which provide the learner with explicit instructions to process information, and *system-assigned detached* strategies which guide the learner to select previously acquired processing strategies (Allen and Merrill, 1985).

Based on the individual, and on the content to be learned, IDP chooses an appropriate set of teaching and learning strategies which are expected to trigger the desired cognitive processing in the learner. Moreover, since the learning process is dynamic, and new information about the student can arrive at any time, IDP can dynamically modify the strategies selected as instruction proceeds. These tasks are accomplished in IDP by employing a dynamic planning approach.

Instructional Planning and IDP Design

The part of an ITS that is responsible for representing knowledge about instruction and making pedagogical decisions is usually referred to as the *instructional component*. Pedagogical decision-making is a complex process in which the tutor has to decide *what* to teach, and *how* and *when* to teach, in order to provide the most instructionally effective means to foster learning (Brecht, 1990). The number and complexity of the various decisions that have to be made point to the need to treat the entire process as a *planning* problem (Macmillan and Sleeman, 1987). Hence, the term *instructional planning* is used for the process in which an ITS decides what to do next at each point during an instructional session.

IDP is embedded in the instructional component of SCENT-3 (McCalla et al., 1990), an ITS which provides advice to students learning to program in LISP. In SCENT-3, instructional planning is separated into two phases: *content planning* which decides *what* to teach, and *delivery planning* which decides *how* to deliver the content plan. The content planning phase has been elaborated by Brecht (1990) who has developed a content planner *PEPE*, which generates a content plan based on characteristics of the learner and instructional goals to be accomplished. The content plan is dynamically adjusted during the instructional process. Content planning represents the first step towards individualization. IDP complements *PEPE* by providing an environment, responsive to learners' individual cognitive processing needs, where a learner can process content in a way that facilitates learning.

IDP is responsible for *executing* the content plan generated by *PEPE*. For each concept to be learned, IDP identifies the learning functions that must be engaged, and calls upon its repertoire of teaching and learning strategies to engage these functions. It then generates a delivery plan tailored to the individual learner. This plan is closely monitored by other components of SCENT-3. If the diagnosis component reveals that the student is having problems in processing the content, IDP reacts by altering the strategies used to influence the student's cognitive processing. This process is repeated until the student learns the content, or until the content planner decides that the student should learn something else.

IDP provides instruction to students learning to write recursive LISP functions, in particular, *cdr (tail-end) recursive functions*. Brecht (1990) treats the concept of recursion as a cognitive skill and organizes this skill into three levels: *fact*, *analysis* and *synthesis*. The ultimate goal of IDP is to facilitate the student learning to construct (synthesize) cdr recursive functions. However, the student must first know what a cdr recursive function looks like (fact) and should understand how and why the parts of the function fit together (analysis).

In IDP, fact, analysis and synthesis of cdr recursion are represented as three separate entities with each entity aware of the learning functions that must be engaged. For example, the *encoding* learning function must be engaged when the student is learning a fact. There is also a set of teaching strategies (TS), embedded learning strategies (ELS) and system-assigned detached learning strategies (SADLS) which can be used to engage the required learning functions. Each strategy is associated with a set of learning functions it can engage and is executed by a *strategy controller* which selects appropriate tactics to implement the strategy.

Teaching strategies can directly engage learning functions. For example, a TS to have the student learn the facts of cdr recursion could be "provide several examples to demonstrate the appearance of a cdr recursive function." Embedded and system-assigned learning strategies are particularly important when the student is learning to analyze and synthesize cdr recursive functions. ELS provide instruction, but also direct the student to perform some sort of cognitive processing. For example, an ELS for analysis may present two examples, one buggy and the other correct, and say, "Look at the two functions. How is the recursive call different?"

SADLS assume that the learner has already acquired the necessary learning strategies and guide the learner to select appropriate ones for the current situation. For example, if the student has submitted an incorrect solution without checking it, an appropriate SADLS is to suggest tasks for the student which can highlight the error.

Teaching strategies as well as embedded and system-assigned learning strategies are implemented by various *tactics*. These tactics are low-level procedures primarily responsible for physically imparting knowledge to the student, as dictated by the type of teaching/learning strategy chosen. In the case of ELS and SADLS, tactics also provide directions to the student to perform some sort of cognitive processing.

Various rules control the selection of strategies and tactics in IDP. For example, rules such as "*prefer ELS over TS*" and "*prefer SADLS over ELS*" maximize the learner's participation in the learning process. In order to maintain motivation and make choices when several options are possible, IDP contains rules such as "*prefer student's choice over* *system's choice (with advisement).*" There are also rules which attend to cases where learning functions are not engaged. Two examples are: "*allow a strategy controller to persist in engaging a learning function which was previously diagnosed as being not engaged*" and "*if a student is having problems with SADLS, use ELS or TS.*"

For the purposes of this research it is assumed that the student model in SCENT-3 is extended to include information about learning functions. IDP uses this information (presently as an oracle) to guide the selection of strategies appropriate for the student. For example, if the student model indicates that the *hypothesis generation* learning function was not engaged during analysis, IDP may react by choosing the ELS "present a buggy example and ask the student to trace it with various data." SCENT-3's student history is also important to IDP, since history records the student's tutorial preferences and the context of past instruction. This helps IDP to select strategies appropriate to individual learners. For example, the information that "*student prefers ELS over SADLS*" causes IDP to favour an ELS, even though it was contemplating the use of an SADLS. The context of past instruction allows IDP to repair blocked plans by selecting different sets of tactics, thus providing multiple contexts for the student to encode information (Shuell, 1991).

Adapting Instruction in IDP

A brief example is now provided to illustrate how IDP works. Suppose IDP is triggered by a content plan from *PEPE*, like the one in Figure 1. Introduce is used to motivate and establish appropriate expectations within the learner and is not elaborated in this example.

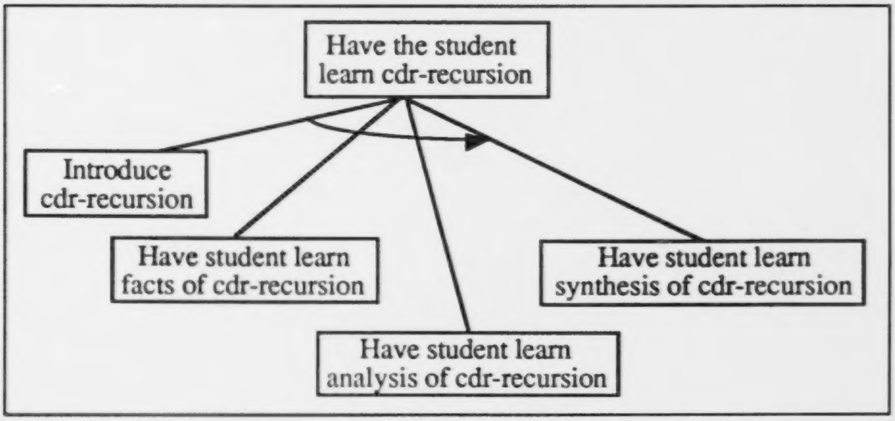

Have the student learn cdr-recursion
Introduce cdr-recursion
Have student learn facts of cdr-recursion
Have student learn synthesis of cdr-recursion
Have student learn analysis of cdr-recursion

Figure 1. A Content Plan

After introducing cdr recursion, IDP plans to impart facts about cdr recursion to the student. It examines the fact entity which reveals that the *encoding* function is to be engaged. IDP then consults the student history to obtain any preferences the student may have. Suppose that the student has had problems with ELS and SADLS in learning other types of recursion.

Selection rules would cause IDP to focus on a TS, such as "provide several examples ..." IDP might then plan to provide two examples and permit the student to request up to three additional examples. IDP would also select tactics to display the examples, with instructions to highlight the important parts of the recursive function (which helps to engage the *encoding* learning function). It also would record in the student history the particular tactics that were selected. After execution of the *display example* tactics, the plan controller allows the student to request more examples. However, after several examples, the plan controller would encourage the student to enter the analysis phase.

In order to accomplish analysis, three learning functions: *encoding*, *hypothesis generation*, and *comparison* must be engaged. IDP makes several strategy selections to engage these functions. Based on the student history, it chooses to first employ a TS to present sample cdr recursive functions and to explain the purpose of each part of the function (to trigger *encoding*). In order to encourage *hypothesis generation* and *comparison*, IDP plans to follow up with an ELS which questions the student about the material presented. Following selection of tactics and initiating the plan, suppose that the diagnostic component revealed that only the *encoding* function had been successfully engaged. IDP would then attempt to engage the *comparison* and *hypothesis generation* learning functions in some other way. IDP would replan, possibly choosing a TS which directly tells the student what the comparison is, and then possibly using an ELS to engage the *hypothesis generation* function. This process continues until the student has engaged the required learning functions. While engaging learning functions does not guarantee successful learning, it is the success measure of the delivery planner. The content planner decides what to teach based on the learning that has occurred.

Synthesis of cdr recursion functions can be taught by providing guided practice for the student, followed by independent synthesis of functions. This is the default order, but can be overridden by the student. Similar to fact and analysis, the plan is formulated and various tactics are carried out to accomplish the selected strategies. The delivery plan for synthesis can interact with plans for fact and analysis. For example, suppose that the student's synthesis of a function is incorrect and diagnosis reveals that the student failed to analyze the solution for correctness. This would trigger an appropriate analysis sub-plan to be formulated which would encourage the student to analyze the solution obtained. Control then returns to the original synthesis plan which continues to provide problems for the student to solve. In this way, IDP is able to adapt instruction to the cognitive processing of individual learners.

Sometimes, IDP may find itself unable to select an appropriate set of strategies, or tactics to implement them. Also, IDP may be unable to formulate an appropriate plan to engage certain learning functions. One option at this stage is to execute the next node in the content plan. Another option is to return control to the content planner, which would react to this blocked learning path by selecting content which appears more appropriate for the learner (Brecht, 1990). This is not very meaningful at the moment, since IDP currently has a narrow range of content knowledge. However, this interaction will become more important when IDP is extended to handle more concepts in LISP programming.

Conclusions

The IDP model has not yet been fully tested and evaluated. However, efforts to do so are currently underway. Although no claims about its usefulness can be made at this time, research to date indicates that it is fruitful and desirable to plan instruction based on an explicit cognitive model of instruction. Theoretically, the instruction provided in IDP is educationally sound, since the cognitive model is strongly supported by much research in education. This contrasts sharply with several approaches to building instructional computing systems where knowledge is often imparted in *ad hoc* ways.

One limitation of IDP is that it is not a complete working system. It has been built primarily to investigate the process of incorporating a cognitive model of instruction in an ITS. To this end, it was necessary to assume that various components of SCENT-3 would provide information when and as needed. For example, it is assumed that a diagnostic component exists which provides information about the learning functions engaged within the learner. This diagnosis is an extremely difficult task (Shuell, 1991). Nevertheless, it is a vital requirement for IDP since instruction is provided according to the diagnostic information obtained. This problem of cognitive diagnosis is currently being tackled by researchers such as Winne (1991) and Bhuiyan et al. (1991).

One thorny issue arising out of this research is the lack of computational precision in the learning functions. There are complex inter-relationships between the functions and some are recursive (Shuell, 1991). Because of this, only a few of Shuell's learning functions are presently dealt with in IDP. However, a complete model of instruction should involve all of the learning functions interacting in predictable ways. This will require describing the learning functions in greater detail and precision.

Another issue for future research is domain-independence. While IDP presently focusses on the domain of recursion in LISP, many of the knowledge representation issues are the same for other procedural problem-solving domains. Teaching and learning strategies are relatively domain-independent. On the other hand, the tactics selected to implement strategies are, by necessity, domain-dependent. Thus a subsequent phase of IDP's development is to extend the architecture to other domains. This will increase the usefulness of IDP as well as provide more opportunities to validate the cognitive model it is based on.

Acknowledgements: We would like to thank the Natural Sciences and Engineering Research Council of Canada for their financial support of this research, and the Canadian Commonwealth Scholarship Committee for their support of the first author.

References

Allen, B.S. and Merrill, M.D. (1985). System-assigned Strategies and CBI. *Journal of Educational Computing Research* 1(1): 3-21.

Anderson, J.R., Boyle, C.F., Corbett, A.T. and Lewis, M.W. (1990). Cognitive Modelling and Intelligent Tutoring. *Artificial Intelligence* 42(1): 7-49.

Bhuiyan, S., Greer, J.E. and McCalla, G.I. (1991). *Characterizing, Rationalizing, and Reifying Mental Models of Recursion*. ARIES Laboratory Research Report 91-4, University of Saskatchewan.

Brecht (Wasson), B.J. (1990). Determining the Focus of Instruction: Content Planning for Intelligent Tutoring Systems. *PhD. Thesis*, Department of Computational Science, University of Saskatchewan.

Jones, M. (1988). *Instructional Systems Need Instructional Theory: Comments on a Truism*. ARIES Laboratory Research Report 88-9, University of Saskatchewan.

Leinhardt, G., and Greeno, J.D. (1986). The Cognitive Skill of Teaching. *Journal of Educational Psychology* 78(2): 75-95.

Macmillan, S.A. and Sleeman, D.H. (1987). An Architecture for a Self-Improving Instructional Planner for Intelligent Tutoring Systems. *Computational Intelligence* 3(1): 17-27.

McCalla, G.I., Greer, J.E. and the SCENT Research Team (1990). SCENT-3: An Architecture for Intelligent Advising in Problem-Solving Domains. In C. Frasson and G. Gauthier (Eds.), *Intelligent Tutoring Systems: At the Crossroads of Artificial Intelligence and Education*, Ablex.

Shuell, T.J. (1991). Designing Instructional Computer Systems for Meaningful Learning. In P. Winne and M. Jones (Eds.), *Adaptive Learning Environments*, Springer Verlag.

Weinstein, C.E., Goetz, E.T. and Alexander, P.A. (Eds.) (1988). *Learning and Study Strategies: Issues in Assessment, Instruction and Evaluation*, Academic Press.

Weinstein, C.E. and Mayer, R.F. (1986). The Teaching of Learning Strategies. In M.C. Wittrock (Ed.), *Handbook of Research on Teaching (3rd Edition)*, Macmillan.

Winne, P.H. (1991). State of the Art Instructional Computing Systems that Afford Instruction and Bootstrap Research. In P. Winne and M. Jones (Eds.), *Adaptive Learning Environments*, Springer Verlag.

Conceptual Restructuring in History of Mechanics as a Model for Learning Mechanics

Ira A. Monarch
LCL, Philosophy Dept, Carnegie Mellon University, Pittsburgh, PA 15213

C. Franklin Boyle
CDEC, Carnegie Mellon University, Pittsburgh, PA 15213

Dewey I. Dykstra, Jr.
Department of Physics, Boise State University, Boise, ID 83725

Abstract: A case has been made for comparing students' preconceptions with conceptual complexes in pre-Newtonian mechanics. It has been further suggested that student preconceptions are as equally resistant to change as the conceptions of Newton's predecessors. To address *conceptual resistance* in students, some have maintained that significant moments of conceptual restructuring in the *construction* of mechanics can be used to achieve better understanding of student constructions. However, only preliminary work has been done to specify the nature and content of conceptual restructuring drawn from history as it is relevant to cognitive models of learning. This paper takes this preliminary work a bit further and describes an episode of conceptual change that can apply in greater detail to the study of such change in students. In particular, the focus is on a well-researched area in the history of mechanics: Galileo's changing conceptions of motion, both with respect to his predecessors and to his own initial views. It is noted that Galileo was never able to specify the relation between weight as measured on a balance and the tendency to move downward. It is proposed, in conclusion, that identifying concepts and problem situations that are important to Galileo's conceptual changes can be useful in understanding conceptual change and *learning* in students.

History of Conceptual Change in Mechanics as Relevant to Learning Mechanics

In the last decade or so, a number of studies have shown that many of the difficulties students have in learning science, more specifically physics and mechanics, are similar in fundamental ways to problems encountered by scientists and philosophers who were instrumental in the development of these fields (Clement, 1983; di Sessa, 1982; Driver and Easley, 1978; McCloskey, 1983). Other studies have suggested that conceptual change in the history of mechanics has important similarities to conceptual change required for learning mechanics (Posner et al., 1982; Nersessian, 1989; Nersessian and Resnick, 1989; Robin and Ohlsson, 1989). These investigations have been responses to: 1) findings that students begin formal physics education with a system of physical conceptions that differ in deeply systematic ways from those of the physicist and thus present a significant obstacle to learning physics (Viennot, 1979; di Sessa, 1983; McDermott, 1984; Halloun and Hestenes, 1987), and 2) the recognition that these pre-existing systems of conceptions must be modified in the course of instruction (Viennot, 1979; Halloun and Hestenes, 1985; McDermott, 1984; Champagne et al., 1980; Minstrell, 1989; Boyle et al., 1990).

Implicit in the realization that conceptual change in science and in science education have important overlaps is the idea that conceptual change, as it has occurred in the history of science, provides a valuable resource for investigating the general issues involved in knowledge restructuring and, in some cases, may even aid in probing student understanding (Nersessian, 1989). In the case of impetus, for example, investigators may not have identified student views as containing a concept of impetus. More importantly, they would not have known to probe students that apply a concept of impetus to projectile motion to determine whether they also apply it to falling bodies (Nersessian and Resnick, 1989; Robin and Ohlsson, 1989) unless they had had available to them historical studies of impetus. However, such research, especially concerning the conceptual path from impetus to inertia, has been dependent on a limited set of historical studies, focusing on a rather small set of problem situations. The aim of this paper is to add to the repertoire of historical studies relevant to cognitive research and to augment the number of problem situations to be considered or at least interpret old problem situations in new ways.

Should Ontogeny Recapitulate Phylogeny?

Some preliminary discussions of the importance of the history of science to science education have come under the rubric of: aOntogeny Recapitulates Phylogeny' (Nersessian, 1989). As in the case of its biological counterpart (Gould, 1977), we must be careful in how we interpret this recapitulation. What aspects of cognitive development in the history of mechanics should be recapitulated in student cognitive development? It is clear that students learning mechanics do not repeat the history of mechanics. It is not just that the social and cultural milieu is different for contemporary students than for past scientists, but certain mathematical techniques such as the use of simple algebraic equations were not available or, if available, not used until well into the 18th century (Hankins, 1970). Galileo's laws, for example, were expressed in geometrical proportions and his figures were all geometrical. It could be argued that it might be beneficial for students to struggle with the mathematical constraints that Galileo worked under, so that they might better appreciate the power and effectiveness of present mathematical techniques. However, such a recommendation points to one of the difficulties of applying recapitulation to pedagogy. How is it decided what to repeat and when? While a detailed answer to this question is not attempted here, it is suggested that the study of *conceptual change* is essential to applying history of science to science education because similar kinds of conceptual change are fundamental to scientific practice and scientific education. This suggestion will be formulated through a discussion of Galileo's construction of a new science of motion.

Studying Galileo's Early Work to Gain Insight into Conceptual Restructuring

If the historical study of conceptual change in mechanics and a cognitive analysis of conceptual change in the classroom are to mutually support each other (Robin and Ohlsson, 1989), studies in the history of science need to be made available that trace traditions of mechanical problems and the various interconnected conceptual responses to them. It might be useful to know the situations and images used, for example, by Galileo because they may be more intuitive to or cohere better with students' views than those presented by post-

Newtonian teachers. Although an impetus-like notion as applied by students to problems involving falling bodies and projectile motion has been studied, the interrelationship of these problems to problems in statics (balance, lever, inclined planes), hydrostatics and the new mechanical problems on inclined planes, especially as set out in Galileo's early work, have not been considered. Even those who have considered Galileo's work, in particular his early impetus theory of projectile motion, as a source of pedagogical insights (McCloskey, 1983; Clement, 1982), have overlooked these conceptual and domain-forming contexts (McCloskey, 1983; Clement, 1982, 1983; Nersessian, 1989).

It will be argued here that, in many ways, it is Galileo's early work which should be of most interest to science education. It is in this work that Galileo determined many of the problem situations that should be included in the domain of the new science of motion and many of the conceptual tools needed for representing, deducing and explaining the phenomena of motion. It is also with respect to this work that the locus of Galilean conceptual change can best be studied both with respect to previous traditions and with respect to his own later work that resulted in conceptual tensions he never resolved. These matters are important for science education because in learning mechanics, students have to *construct* the anew' science of motion all over again. The kinds of reclassification of phenomena and conceptual breakthroughs needed for this construction by students today can be found exemplified in Galileo's early work. In fact, one of the important aims of Galileo's writings, both early and late, is pedagogical (Clement, 1983) and the key to his pedagogy is to make his readers aware of the enormous conceptual breakthroughs necessary to establish the new science of motion, his critique of Aristotle being emblematic in this regard.

Conceptual Restructuring in Galileo's Early Work: Reclassification

One of the aims of Galileo's early work, *On Motion*, is to explain how all *natural* motion is the result of essential *heaviness*. He accepts, at this point, the Aristotelian notion of natural place that bodies move to their proper places as indicative of their relative heaviness—the more heavy moving closer to the center of the universe than the less heavy. Moreover bodies move *upward* because of their relative lightness or *lesser heaviness*, or, in other words, free upward motion results from a body being immersed in a heavier medium. In the case of natural motion, then, the relative heaviness of the medium through which the motion takes place must be considered as well as the relative heaviness of the moving body itself.

These considerations prepare the way for Galileo to connect Aristotle's dynamic notion of natural motion to his (Galileo's) extension of Archimedes' hydrostatic analyses. Galileo asks why a stone cast into the sea moves downward naturally, even though the sum total of water of the sea is far heavier than the stone? His answer is that the stone is heavier than the water in the sea and will move downward in it because a quantity of water that is *equal in volume* to the volume of the stone is *lighter* than the stone. But why, asks Galileo, should the physical process of natural downward motion depend on the two volumes being equal? His answer is quite indicative of his approach to the new science of motion, his main topic in *On Motion* (p.17). It depends on the notion that a body immersed in a medium *displaces* a volume of the medium equal to the volume of the body. If the weight of the medium is the same as the body, the volume of the medium displaced will press upward to displace the immersed body as much as the body will press downward, and there will be no motion. If the medium is heavier, it will push a body of equal volume upward and, if lighter, the body will move downward since it can neither be at rest nor move upward.

Not stopping at hydrostatics, Galileo further extends the new science of motion to the science of weights on a balance on the basis of the analogy between bodies that move naturally and the statics of weights on a balance. "The body moving naturally plays the role of one weight in the balance, and a volume of the medium equal to the volume of the moving body represents the other weight in the balance" (*On Motion* p.23).

It is important to see that this analogy is really a way of bringing a variety of mechanical situations that, for the most part, had been treated separately up to Galileo's time under one perspective and to treat them uniformly. This is an important step in *creating* the new science of motion and it may be an important consideration in *learning* mechanics as well. The natural motion of falling bodies had not been classifiable with bodies floating or rising in a medium. Nor were freely falling bodies and bodies on a balance seen in the same light. But Galileo was recommending that these classifications should be changed, that *unconstrained* and *constrained* motion should be classified under one perspective. Moreover, he was extending *explanations* based on the notion of equilibrium to freely falling bodies to supplement explanations based on the tendency to move toward the appropriate natural place. This reclassifying of situations and explanations as appropriate to new problems is an important part of Galileo's conceptual restructuring and may be the sort of conceptual change students need to undergo in learning mechanics. Conceptual change is not just a matter of finding a place in a conceptual scheme for a few new concepts like inertia, mass and gravity, but also of interlinking these concepts in a variety of ways to reclassify situations (Boyle et al., 1990).

Conceptual Restructuring: Articulating Principles and Applying Them

The grouping of freely falling bodies with bodies floating or not in water and with bodies on a balance enabled Galileo to produce a quite devastating critique of Aristotle's mechanics. Galileo shows that Aristotle's claim that larger bodies made of the same material move more swiftly is incorrect. Given Galileo's application of statics to freely falling bodies, he can show that bodies of the *same kind*, even if one is larger than another, still move with the same speed because they must raise a larger amount of the medium they pass through. Further, if two bodies of *different kind*, one of which moves faster than the other, are bound together, the resulting body will not move faster than the original faster one, as Aristotle believed, but more slowly. Similarly, if two equal bodies that move with equal motion are joined together, their motion is not doubled or even increased as Aristotle held.

Galileo distills these insights, in particular, that bodies moving naturally can be reduced to the weights of a balance, into the following principle: *A body moves downward with the same force with which it resists being drawn upward.* He then infers that a body moves downward with a force equal to the weight by which it exceeds the weight of the medium, which latter weight is also relative to the medium in which it is weighed. With these results, Galileo introduces another situation to be included in the domain of the new science—bodies moving on *inclined planes*. It is in this context that Galileo articulates another principle, one which can be seen as an important step toward his principle of circular inertia.

Galileo's inclined plane problem can be divided into two questions: 1) Why do bodies move more swiftly along planes that are more steeply inclined? and 2) What is the ratio of the motions along differently inclined planes? Galileo maintains that the answers to these questions depend on the principle of statics formulated above.

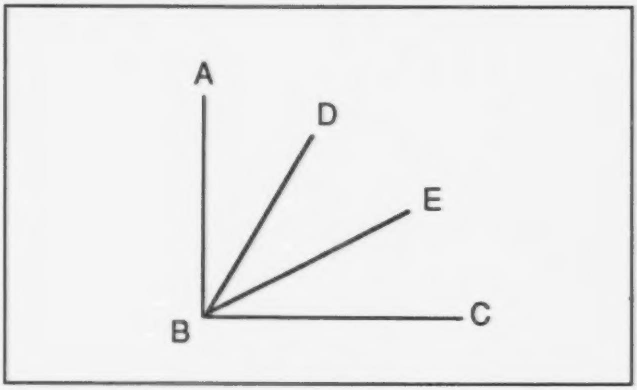


Figure 1.

According to this principle and given that speed is identified with force of descent, it can be determined how much greater the force of descent is along BD in Figure 1 (Wisan, 1974) than along BE when it is known how much more force is required to draw the same body up along BD than along BE. This, in turn, can be known if the ratio of the effective weights of the body on the respective planes is known.

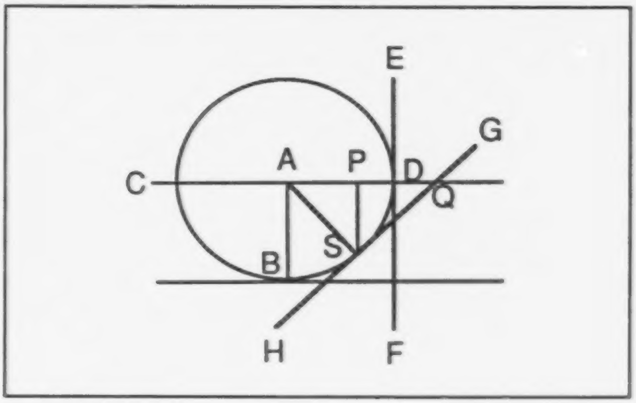


Figure 2.

Given these determinations, Galileo reduces the problem to the consideration of a balance CD in Figure 2 (Wisan 1974) with center at A which has equal weights at C and D. Supposing that the arm AD rotates about A to the point S, the body *at* S would be descending as though along line GH, tangent to the circle at S. The ratio of the weights of the same body at D and S will give the ratio of the speeds along EF and GH. From the principle of the bent lever, elementary geometry and supposing S as though suspended from P, Galileo derives that the ratio of the motions along EF and GH is as QS is to SP or as the length of inclined descent is to the length of the vertical height of that descent. He then concludes:

> the same weight can be drawn up an inclined plane with less force than vertically, in proportion as the vertical ascent is smaller than the oblique. Consequently, the same heavy body will descend vertically with greater force than on an inclined plane in proportion as the length of the descent on the incline is greater than the vertical fall (*On Motion* p. 65).

Galileo will generalize this result to ratios of speeds along planes of any inclination with the same vertical height as being inversely proportional to the lengths of the planes. But first he introduces what has been called his *proto-inertial principle* (Wisan, 1974). He notes that the above proof must be understood on the assumption that both the plane and the moving body are perfectly smooth and hard, that is, incorporeal, or at least carefully smoothed so as to avoid accidental resistance. Under such ideal conditions, Galileo maintains, "any body on a plane parallel to the horizon will be moved by the very smallest force, indeed, by a force less than any given force" (*On Motion* p.65-66).

Galileo acknowledges a little further on that it may be impossible to find such bodies, free from all external resistance. In addition, he also notes that a plane cannot actually be parallel to the horizon, since the surface of the earth is spherical, and a plane cannot be parallel to such a surface (*On Motion* p.68). But then he begins to move in another direction and asks whether circular motion is natural or forced. Since natural motion is motion toward a body's natural place and forced motion is a receding from a body's natural place, a sphere rotating about the center of the universe moves with a motion that is neither natural nor forced. This leads Galileo to ask whether, after receiving a start of motion from an external mover, a body in such a situation would move perpetually or not. This question is never answered in *On Motion*. However, the question has opened up the possibility of a circular inertial motion.

From Proto-Inertia to Circular Inertia: A Possible Conceptual Transition for Students Learning Mechanics

The proto-inertial principle arose for Galileo in the context of his bringing together freely falling bodies, rising and sinking bodies, the behavior of bodies on a balance and bodies moving on inclined planes, that is, bringing static considerations to bear on situations usually thought of dynamically or kinematically. This principle is especially interesting because students who have been introduced to the same range of mechanical and explanatory possibilities might also construct it, especially in *transition* to an inertial concept in less incipient form. In what follows, it will be useful to show: 1) how the conceptual context for Galileo's proto-inertia changed to include circular inertia, a concept very close to, but importantly different from, Newtonian inertia, and 2) how certain deficiencies in Galileo's new conceptual scheme raised an unfathomable problem for him. Identifying such problems and their conceptual contexts may be an effective precondition for teaching mechanical concepts–in this case not only the concept of inertia, but, as will become apparent, the concept of mass as well.

Crucial for understanding Galileo's proto-inertia and the transition to circular inertia is his treatment of projectile motion in *On Motion*. Galileo's point of departure is the Aristotlean view that a mover must be in contact with a moving body. After rejecting Aristotle's view that projectiles are somehow moved by the air, Galileo questions whether a motive force can be impressed by the projector on the projectile? His answer again appeals to his newly constructed domain of motion with its reconfigured view of heaviness, of superimposing statics onto dynamics. The motive force impressed *takes away heaviness* when the body is hurled upward, and *takes away lightness* when the body is hurled downward. In the case of a stone becoming light by being hurled upward, for example:

> its natural and intrinsic weight is lost in the same way as when it is placed in media heavier than itself. For a stone which floats, let us say, in mercury and does not sink, loses all its weight; indeed, it loses weight and assumes lightness to such an extent that it promptly resists even a great deal of weight that comes to it externally—e.g., if one tries to press it down. Wood, too, becomes so light in water that it cannot be kept down except by force. And yet, neither the stone nor the wood loses its natural weight, but, on being taken from those heavier media, they both resume their proper weight. In the same way, a projectile, when freed from the projecting force, manifests, by descending, its true and intrinsic weight (*On Motion* p. 81).

Galileo's notion of impressed force or impetus is thus somewhat different from the medieval conception that has been compared to student conceptions recently (Nersessian and Resnick, 1989; Robin and Ohlsson, 1989), notably in its connection to statics.

Conceptual Tension in Galileo: Prelude to Effective Learning

In the new conceptual context developing after *On Motion* and before Galileo's major works, the *Dialogue Concerning The Two Chief World Systems* and the *Two New Sciences,* Galileo drops the idea of a gradually decaying force impressed upon a body. Instead of force being imparted to a body, it is motion which, under ideal conditions, is permanently maintained, whatever its direction. Such motion is permanently preserved unless a motion in the opposite direction is also imparted to the body. In situations where Galileo had applied the concept of impressed force, he was now applying his concept of inertia, which he interprets as necessarily circular on the surface of the earth. He was able to make this substitution on the basis of another principle conceptually linked to the principle of circular inertia.

In the case where a body acquires a given velocity through natural fall and is then diverted upward along an inclined plane, it will decelerate in this upward motion because it has a natural tendency to accelerate downward in the opposite direction. This deceleration presupposes that all components of any given motion are preserved regardless of their direction and that for a given complex motion one can determine its future trajectory by adding components that comprise it, assuming that each of them acts as if the others do not exist. What is being presupposed has been called Galileo's superposition principle (Prudovsky, 1989). This principle, together with his inertia principle, enables Galileo to represent projectile motion without appealing to a gradually decaying impressed force. For example, if a body is moving uniformly on a frictionless plane and passes over the edge of the plane, it will acquire, in addition

> to its previous equable and indelible motion that downward tendency which it has from its own heaviness. Thus there emerges a certain motion, compounded from equable horizontal and from naturally accelerated downward [motion]. (*Two New Sciences*, p. 217.)

Galileo's shift away from impressed force also required a new explanation of the acceleration of falling bodies. In the earlier context, he had appealed to the removal of a resisting factor which leaves behind an ever-weakening impressed force that continuously increases the velocity of the falling body. In the newly developing context, Galileo generalizes his previous view of bodies falling with equal speed to *all* bodies, regardless of either absolute weight or specific gravity. This means that neither specific gravity nor heaviness, as ascertained by weight in a balance, determines speed of fall. But what, then, does determine speed or acceleration of fall and what is the relation between weight measured on a balance and the tendency to move downward, either vertically or along an inclined plane? Without Newton's concepts of mass and force, in particular gravitational force, this was, for Galileo, an unfathomable problem (Wisan, 1984). Galileo was caught between the conceptual innovations he had made in grafting static considerations to freely falling bodies on the one hand and the new discoveries he was making concerning inertia, superposition, acceleration and all bodies falling at equal speeds on the other.

Conclusion

Such problems may be typical in learning situations requiring conceptual change. In our view, such learning occurs when *knowledge* is *constructed* by the individual. The goal of physics teaching should be to structure classroom interaction to facilitate students developing Newtonian conceptions for themselves. But teachers are in a difficult position. They must be good anthropologists as well as physicists. They must be able to understand student conceptions as well as Newtonian ones. Just as important, they must be able to envision transitional knowledge states which are more viable than initial student ones but less viable than those of the Newtonian physicist. In a sense, this is what Galileo's work provides. An educational process which leads students into the sort of conceptual impasses described for Galileo above and then out of them might be one way to facilitate students developing Newtonian conceptions for themselves. The precise ways in which such a process would draw on Galileo's conceptual development is a topic for further research. It is hoped that this paper has shown that Galileo's concetual development might be an important source for mechanics education.

References

Boyle, C.F., Dykstra, D.I., and Monarch, I.A., Using Knowledge Representation to Study Conceptual Change in Students for Teaching Physics, *Proc of 12th Ann Onf Cog Sci Soc*, Cambridge, 1990

Champagne, A.B., Klopfer, L.E. and Anderson, J.H. Factors Influencing the Learning of Classical Mechanics. *American Journal of Physics* 48(12), 1074(1980)

Clagett, M.. *The Science of Mechanics in the Middle Ages*, University of Wisconsin Press (1959)

Clement, J. Students Preconceptions in Introductory Mechanics. *American Journal of Physics* 50(1), 66(1982)

Clement, J. A Conceptual Model Discussed by Galileo and Used Intuitively by Physics Students, *Mental Models*, Gentner, D. and Stevens, A.L. (Eds.), Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 325(1983)

di Sessa, A.A., Unlearning Aristotelian Physics: A Study in Knowledge-Based Learning, *Cognitive Science*, 6, 37-75, (1982) di Sessa, A.A., Phenomenology and the Evolution of Intuition. *Mental Models*, Gentner, D. and Stevens, A.L. (Eds.), Lawrence Erlbaum Associates, Inc., Hillsdale, New Jersey, 15(1983)

Driver, R. and Easley, J. Pupils and Paradigms: A Review of Literature Related to Concept Development in Adolescent Science Students. *Studies in Science Education* 5, 61(1978)

Galilei, G., *Dialogue Concerning Two Chief World Systems*, S. Drake (trans.), University of California Press, 1632 (1962)

Galilei, G., On Motion, in I. E. Drabkin (trans), *Galileo on Motion and on Mechanics*, University of Wisconsin Press, 1590 (1960)

Galilei, G., *Two New Sciences*, S. Drake (trans.), University of Wisconsin Press, 1638 (1974)

Gould, S. J., Ontogeny and Phylogeny. Harvard University Press, Cambridge, Mass (1977)

Halloun, I.A. and Hestenes, D. Modeling Instruction in Mechanics. *American Journal of Physics* 55(5), 455(1987)

Hewson, P.W. Epistemological Commitments in the Learning of Science: Examples from Dynamics. *European Journal of Science Education* 7, 163(1985b)

Hankins, T.L., *Jean d'Alembert: Science and Enlightment*, Clarendon Press, Oxford, (1970)

Koyre, A. *Metaphysics and Measurements*. Harvard University Press, Cambridge, MA (1968)

McCloskey, M. and Kohl, D. Naive Physics: The Curvilinear Impetus Principle and Its Role in Interactions with Moving Objects. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 9, 146(1983)

McDermott, L.C. Research on Conceptual Understanding in Mechanics. *Physics Today* 37(7), 24(1984)

Minstrell, J. A. Teaching Science for Understanding. *Toward the Thinking Curriculum: Current Cognitive Research*, Resnick, L. and Klopfer, L. (eds), Alexandria, VA, Association for Supervision and Curriculum Development, 129 (1989).

Moody, E.A., The Dynamics of the Leaning Tower Experiment (I & II), *Journal of the History of Ideas*, 12, 163-93, 375-422 (1951)

Nersessian, N. J. and Resnick, L.B., Comparing Historical and Intuitive Explanations of Motion Does "Naive Physics" Have a Structure?, *Proc of 11th Ann Onf Cog Sci Soc*, Ann Arbor, Aug, (1989)

Nersessian, N. J., Conceptual Change in Science and Science Education. *Synthese* **80**, 163(1989)

Posner, G.J., Strike, K.A., Hewson, P.W. and Gertzog, W.A. Accomodation of a Scientific Conception: Toward a Theory of Conceptual Change. *Science Education* 66, 211(1982)

Prudovsky, G., The Confirmation of the Superposition Principle: On the Role of a Constructive Thought Experiment in Galileo's *Discorsi*, *Stud. Hist. Phil. Sci.*, 20:4, 453-469, (1989)

Robin, N. and Ohlsson, S., Imetus Then and Now: A Detailed Comparison between Jean Buridan and a Single Contemorary Subject, *Science Teaching Conference*, Florida State University, (1989)

Viennot, L. Spontaneous Reasoning in Elementary Dynamics. *European Journal of Science Education* 1, 205(1979)

Westfall, R.S, *The Concept of Force in Newton's Physics*, (1978)

Wisan, W.L., The New Science of Motion: A Study of Galileo's De Motu Locali, *Arch Hist Exact Sci*, 13, (1974)

Wisan, W.L., Galileo and the Process of Scientific Creation, *ISIS*, 75: 269-286, (1984)

Cascaded Problem Sets:
A Model for Learner-Driven Instruction

David Neves, Robert Campbell, Matthew Greising, and Ken Greenlee
The Institute for the Learning Sciences
Northwestern University, Evanston, IL 60201 USA

Abstract: It is traditional in school and industrial education to teach simple principles first, then gradually build up the complexity of the materials until the student is able to solve real problems within a particular domain. We argue that this approach is not only dry and unmotivating to the student but also can puzzle the student as to the relevance to introductory material. Instead we propose that instruction be turned around—that real problems be presented to students and that students be given component problems should these problems prove too difficult.

We start by presenting the student with the problem of keeping the books for a small company. This gets the student interested by presenting a real life problem, not a small artificial problem that they cannot relate to. Should students have difficulty with this problem they can choose to work on related problems that are components of the original problem. As they cascade to simpler and simpler problems they arrive at a problem or instruction that they will understand.

Cascaded Problem Sets works because more able students are not bored with material they already know and less able students are able to get any amount of help that they need.

Let's follow a student sitting through a course on introductory accounting. Near the beginning of the course the instructor covers the many basic accounting terms. The student is bored because he has had some previous accounting experience. Later on, the instructor describes accounting principles that are new to the student. The student understands the definitions somewhat but doesn't understand how knowing these principles will help him do an accounting problem.

We illustrate two major instructional problems above. First, the student is presented with material that he already knows. Not only does this waste the student's time, but also may cause him to tune out and miss new material. Second, even if the student is presented with new material, that material needs to be placed in a context. The student may understand it, but not where it fits in. Our project offers a solution to these two problems. Before discussing it, we first discuss the design of traditional instruction.

Instructional Hierarchies

Many instructional theories are based on Gagne's notion of learning hierarchies derived through task analysis (Reigeluth, 1987). Gagne defines two classes of task analysis:

procedural task analysis and learning-task analysis. A procedural task analysis describes the steps required in performing a certain task or skill. A learning-task analysis describes theskills it takes to perform one of the steps identified in the procedural task analysis. The basic premise is that the skills required to perform a procedural step of the final performance task may be decomposed into a set of component skills. These component skills are in turn decomposed into lower-level component skills. This process continues until skills are decomposed to a level such that the next lowest level represents skills and knowledge that the learner has already mastered. This hierarchy of component skills is called a learning hierarchy (Gagne, Briggs, and Wager, 1988).

Once the hierarchy is developed the instructional designer must provide a sequence for the instruction (Gagne, Briggs, and Wager, 1988). Most instructional theories prescribe sequencing strategies that generally go from simple to complex. This means that, in many cases, the bottom level of the learning hierarchy defines the entry point for the learner; and in almost all cases, the learner is locked into a predetermined sequence of instruction that will typically cover every component skill on the learning hierarchy.

The problem with forcing all learners through a predetermined sequence of instruction is that it ignores the individuality of each learner. This approach assumes that all learners are basically the same–the same skills, the same knowledge, the same learning ability. It also tends to ignore other significant and highly individual factors that contribute to learning, such as the learner's attitudes, interests, and level of motivation.

Cascaded Problem Sets

Our approach to instruction uses a learning hierarchy, but the delivery to the learner is quite different than the traditional linear form of delivery. We begin by presenting the learner with the problem at the *top* of the hierarchy. This sets the context for instruction that will follow and also allows capable learners to quickly pass out of instruction. If learners are able to do the *top level problem* (the problem at the top of the learning hierarchy) there is no need to present them with simpler problems since solving the top level problem represents mastery of the material. We allow less knowledgeable learners to cascade to component problems and instruction until their level of expertise is reached. From there the learners can work their way back up to the top level problem. We call this hierarchical representation and presentation of problems and instruction, *Cascaded Problem Sets*.

Cascade Problem Sets addresses the two instructional failures we mentioned at the start of the paper. First, because we start with a challenging problem we do not bore the learners who have some knowledge of the domain. We do not force-feed them with information they already know. Second, all component problems and instruction are presented in the context of the top level problem. The learner understands both *why* he is presented with a new problem (because it will help him solve the top level problem) and *how* it will help him (because it is a component of the top level problem).

We will next describe how we break down a problem and how we present this decomposition to the learner.

Decomposition

There are a number of methods that can be used in the decomposition of a cascaded problem set. Due to the procedural nature of our content domain (basic accounting), the approach we take is a procedural task analysis.

We begin by breaking down the top level problem into high-level, procedural steps. Each procedural step is in turn decomposed into component procedural steps. We continue this process until we arrive at a point where the components are more conveniently represented as pieces of information required to complete the parent node as opposed to procedural steps of that parent. At this point, we identify the information pieces required to do these lower-level parent nodes. The result is two types of nodes in the problem set—problem nodes (component procedural steps) and instruction nodes (information pieces).

Here is a brief example taken from our cascaded problem set for accounting. The task for one of the problems is to record debits and credits to the accounts affected by a set of business transactions. The procedural steps involved in this are:

1. Determine which accounts are affected by each transaction, and
2. Make debit and credit entries to the appropriate accounts.

So at the next level down in the problem set, there are two problem nodes that correspond to these steps. Now, consider the second of these problem nodes–making the debit and credit entries to the appropriate accounts. This has three procedural steps:

1. Determining whether each account needs to be increased or decreased,
2. Determining what account category each account can be classified as, and
3. Making a debit or credit entry to each account by applying debiting and crediting rules for account categories.

The next level of the problem set, then, consists of three problem nodes that map to these three procedural steps. Consider the second of these problem nodes–determine what account category each account can be classified as. In order to complete this procedural step, learners need to understand the different account categories and the accounts that are classified under each of them. This really is not a lower-level, component procedural step, but rather a concept or piece of information that must be learned in order to complete the procedural step of classifying an account by its account category. So, at this point, rather than a problem node, we provide an instruction node that covers this information.

The benefit of this approach is that the hierarchy is very conducive to task-driven navigation. The learner is focused on completing the top level problem. When he cascades to lower levels on the hierarchy, it is only to visit those nodes that will help him to solve elements of the top level problem. The lower-level problem nodes are particularly important for two reasons:

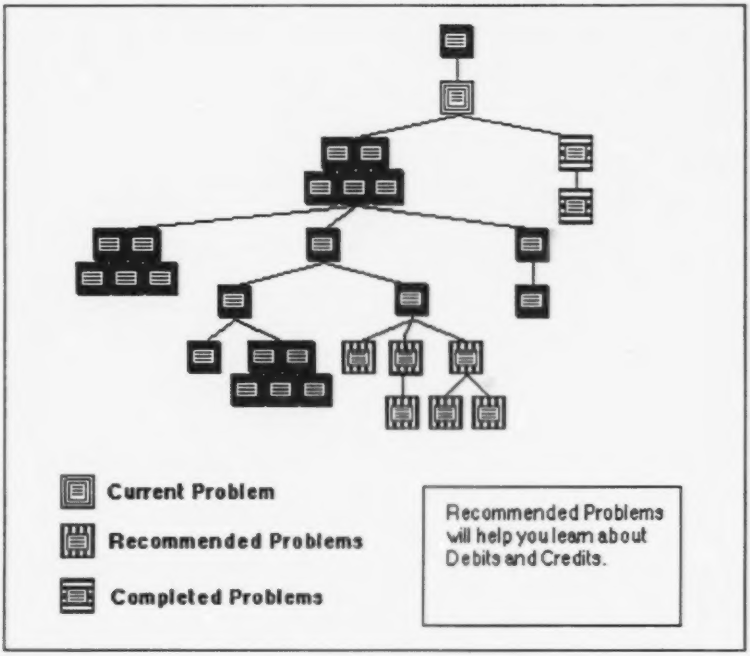
1. They simplify the top level problem, and,
2. They define what the learner does and does not know (i.e., they serve as indices to precisely those instruction nodes that each individual learner must cover to be able to complete the top level problem).

The Interface

The power of our program lies in its ability to provide the learner with exercises and training which are particularly suited to the learner's interests and abilities. The interface encourages exploration and assists the learner in finding information. At any time during the lesson, the learner can ask specific questions or change to any problem or instruction node.

Questions can be asked by clicking on one of the buttons which are located on the edge of the screen. These buttons are derived from the button set developed at *The Institute for the Learning Sciences* (Jona, Bell, & Birnbaum, 1991) with additions for the accounting domain. The buttons allow the learner to request more information without having to type a query. For example, the buttons in the accounting lesson allow the learner to access a map of all the nodes in the problem set (*Map Screen* button), check on their progress through a particular problem (*Check Answer* button), get help with a problem or the interface (*Help* button), or view an explanation of how the current problem relates to the top level problem (*What's the Point?* button).

In addition to asking questions, the learner can select a different problem or browse the entire problem set by referencing the *Map Screen* (see Figure 1). The Map Screen is a graphic representation of the lesson's problem set and an important navigational aid to the learner. Each node, represented as a small icon, is shown with a link to its components problems or instruction. By color coding the icons shown on this screen, it is possible to convey several important pieces of information. This information includes orientation (Where am I? What should I do now?), navigation (Where can I go from here?), progress (How much material have I seen? How much more material is there for me to see?), and control (What are my options at this point?). From the Map Screen, the learner can see a description of any node by pointing to its icon with the mouse. The learner can use these explanations to decide which problem or instruction nodes might be interesting. Clicking on one of the icons will take the learner directly to that node.



Recommended Problems will help you learn about Debits and Credits.

Figure 1. The Map Screen

Another powerful feature of our program is the ability to respond incrementally as the learner attempts to complete a problem and to suggest appropriate remedial material. The *Check Answer* button will review the part of the problem which has been completed and indicate which portions are incorrect. In the accounting problems for example, the learner may be expected to enter information on an ledger sheet. Clicking on *Check Answer* will highlight any entries which are incorrect.

Once the program has evaluated the learner's errors, it is possible to direct the learner to one or more nodes in the problem set which will provide relevant information. For example, if the learner appears to be confusing the concepts of debits and credits, the program can recommend problems and instruction which focus on that distinction. Recommended problems are marked on the Map Screen (see Figure 1). From there, the learner may select one of the recommended problems or the learner may choose to ignore the program's recommendations and browse through other parts of the problem set.

Testing of Prototype

We have finished our first prototype and are beginning evaluation of the program. The key things we look for in the evaluation are:

> Does it allow learners who already know something about the domain to quickly pass through the instruction?
>
> Does it give the help to learners who have no knowledge about the domain?

We have taken a small part of the problem set (15 problem and instruction nodes) and have shown it to five recent hires from Andersen Consulting--most of whom have had some accounting experience. For all but one of the participants the program was used to brush up on accounting skills they had acquired some time ago. These participants ignored the intermediate problems. They worked solely on the top level problem. Check Answer feedback lead them to instruction nodes and they spent most of their time going between instruction and the top level problem.

One of the participants was new to accounting. He got an early and buggy prototype of the program and seemed to flounder in the most difficult branches of the problem tree. We guided him to a simpler problem near the bottom of the problem set where he was more successful.

Future Work

We are continuing to test our prototype. Here are some possible changes for the next version of the program.

We do not have adequate help for novices. If learners know something about the domain they can try the top level problem and get feedback which will lead them to other problems or instruction. It doesn't make sense to have a complete novice attempt the top level problem. We are considering a mechanism that will quickly guide a novice to a much lower level. However it is still important that the novice first understand what the top level problem is about since this will provide context as he views other nodes in the problem set.

Our initial prototype takes a very pragmatic approach to teaching. Only if a problem or instruction is absolutely necessary to solving the top level problem do we include it somewhere in the hierarchy. This is appropriate for learners who want only enough information to complete the top level problem. However, there are some learners who want a deeper understanding of the subject they are studying. We need a way to present these learners with more conceptual explanations when they want them. One possible solution is use an *Ask Tom* type conceptual graph (Schank, Ferguson, Birnbaum, Barger, and Greising, 1991) representing connections to related concepts (such as background, example or counter examples, etc.) that the learner would like to follow up on.

Conclusion

When first viewed, the Cascaded Problem Set method seems to frustrate, rather than teach the learner. After all, the learner will no doubt be presented with something that he cannot solve. Frustration really comes about when a person is given an inscrutable problem and then gets no help with it and can not progress toward the goal. When Cascaded Problems Sets works correctly, the learner can get help, either by viewing a component problem or appropriate instruction.

A Cascaded Problem Sets approach in schools would help to fight the massive amount of repetition that one finds there. Cascaded Problem Sets seems specially well suited for industrial training situations where the learning population is very diverse in their experience and in performance support systems for on the job training.

The Institute for the Learning Sciences was established in 1989 with the support of Andersen Consulting, part of The Arthur Andersen Worldwide Organization. The Institute receives additional support from Ameritech, an Institute Partner, and from IBM.

Acknowledgements: We thank Dr. Ira Hyman for his assistance in the testing of the prototype.

References

Gagne, R, Briggs, L, Wager, W. (1988). *Principles of Instructional Design.* New York: Holt Rinehart and Winston.

Jona, M., Bell, B., & Birnbaum, L. (1991). *Button Theory: A taxonomic framework for student-teacher interaction in computer-based learning environments*. Technical report, Northwestern University, The Institute for the Learning Sciences, Evanston, IL, 1991.

Reigeluth, C. (Ed.) (1987). *Instructional Theories in Action*. Hillsdale, NJ: Erlbaum.

Schank, R., Ferguson, W., Birnbaum, L., Barger, J., and Greising, M. *ASK TOM: An experimental interface for video case libraries*. Technical report 10, Northwestern University, The Institute for the Learning Sciences, Evanston, IL,1991.

Computer Support for School Work

Denis Newman
Bolt, Beranek and Newman, Inc.
10 Moulton St., Cambridge, MA 02138, USA

Abstract: Project-based school work calls for a different way of organizing instruction. This paper contrasts the organizational impact of two technology systems in terms of the physical location in the school, the curriculum/assessment system and how time is scheduled. Considered first is a class of computer systems designed to fit in well with the traditional organization of school work. In contrast, an environment called Earth Lab and its approach to supporting project work is described. The importance of addressing the organizational dimension of school work rather than just the content is illustrated.

We have begun developing computer systems to support project-based school work. The differences between project work and traditional school work is a central concern for those involved in the school restructuring movement which promotes changes in the teacher-student relationships, the timeframe of school tasks, and the collaborative work among the students and among teachers (Carnegie Forum, 1986; Sizer, 1984). The introduction of project work involves more than just changing the *content* of instruction from "basic" to "higher-order thinking" skills. A shift is needed in the organization of instruction. School work may begin more to resemble an apprenticeship (Lave & Wenger, 1990; Collins, Brown & Newman, 1989) with the relationship of the computer to the work also changing (Newman, 1990a; in press).

This paper describes the work of the Earth Lab project which has developed a local area network system for schools and conducted formative research on its functioning in several schools over the last four years (Newman, 1990a; Newman, et al., 1989). A contrast will be made between Earth Lab and more traditional approaches to the use of instructional technology, including CAI and intelligent tutoring systems, with respect to the organization of the school work implied by the two models. Our contrast will explore the dimensions of (1) the physical location and distribution of hardware in the school building, (2) the curriculum content, organization and assessment, (3) the timeframe of the school tasks in relation to the scheduling. School technology systems designed to support project-oriented school work must provide greater seamlessness among physical and logical contexts, group work support, and communication and data-sharing capabilities. The approach to assessment also necessarily changes. While traditional CAI and intelligent tutoring systems make use of a model of the content of instruction and/or a strictly defined sequence of tasks, project based approaches depend on tools that reflect progress over an extended period and that provide opportunities for performances that both teachers and students can reflect on. This paper begins with the contrasting case, a class of computer systems that are designed to fit into the structure of traditional school work.

Systems Designed for Traditional School Work

In the last few years, a product category has emerged that makes it quite easy for schools to acquire and use computer technology without major changes in their goals or organization. A report by the EPIE Institute (Sherry, 1990), an independent consumer information organization, provides a detailed catalog of this class of school technology called integrated learning systems (ILS), or more recently, integrated instructional systems (IIS). These systems typically consist of networked personal computers with a file server that both delivers programs to the individual workstations and keeps records on the progress of individual students. The ILS software (or "courseware") covers most of the school curriculum—thus the term "integrated." These systems are growing in popularity according to the EPIE report which estimates that sales doubled in 1989. The ILSs can be considered a successful implementation of computers: one that serves a perceived need and fits in well into the practices of the traditional school. We can examine three prominent features of ILS use: location, curriculum, and the timeframe of the ILS tasks.

Location

The EPIE report notes that it is an all but universal practice to put the ILS in a lab. Although there is nothing inherent in the technology that requires this configuration, ILSs are sold as "labs" in units of about 30 which is sufficient to simultaneously accommodate a whole class assuming that students will be working at the computers individually. In many ILSs, students also wear headsets as part of their computer interactions so are further removed from peer interaction.

Having the ILS centralized in a lab means that the system can be more easily managed by a school computer lab teacher, or non-teaching paraprofessional. For a school starting with a low level of expertise, the centralization reduces the technical and subject matter training costs that would otherwise be necessary. It is also argued that having students occupied with computer tutorials offers teachers an opportunity for individualized dialogue with students who are often ignored in whole class teaching (Schofield et al., 1990). While this is a fundamental change from the traditional classroom lecture style, it does not change the nature of the school work considered in terms of the curriculum goals and class scheduling already in place.

Curriculum and Assessment

Another feature that provides a good fit to traditional school work is the content organization of the courseware. It is straightforward to categorize the courseware into the standard school subjects: math, reading, language arts or science similar to the typical textbooks. Also, like the typical textbook, the content is, for the most part, presented as facts or procedures to be mastered in sequence.

The content and its sequencing is an integral part of the management and evaluation functions. Discrete tasks which result in a single correct answer can be evaluated by the system itself. More open ended tasks requiring, for example, formulation of the problem or research into sources outside the computer, or any kind of free form response cannot be handled by the system. Intelligent tutoring systems (ITSs) are a natural extension of the integrated learning system approach since they also operate best within very confined subject matter domains with similar level of control over student responses.

The division of topics into clearly defined subject areas also eliminates the need for teachers handling different subjects to collaborate or for a teacher in a self-contained classroom to consider the integration of learning across the subjects. In this respect, too, ILSs strongly support work practices in most schools where teachers are not expected to know much about what other teachers are doing.

Timeframe

ILSs are designed on the assumption that each period will be self contained. The tasks do not require preparation prior to the computer lab period and call for a minimum of technical capability on the part of the student. Teachers can thus schedule the use of the computer lab into the preexisting slots in the day. The timeframe assumed by ILSs is thus well suited to the structure of the school day which is broken down into discrete periods. Single tasks, such as an arithmetic problem, can usually be done in a matter of a few minutes. The short tasks of the ILS are also very similar to the kinds of tasks found on the standardized tests. The short, carefully constrained answer slots are ideal for automatic scoring in both instances.

The Earth Lab Project

For the last four years, the Earth Lab project has been designing, implementing, and observing the effects of a local area network system intended to facilitate collaborative project work in elementary school earth science (Newman, 1990a). Our plan was to create a prototype LAN system and demonstrate it in a New York City public school using an earth science curriculum. The pedagogical rationale was that students should engage in scientific work using the technology to communicate and share data.

The school is a public elementary school (grades 3 to 6) located in Central Harlem, New York City. The school population of approximately 700 students is predominantly African-American with a minority of Hispanic and other groups. The school's achievement scores are about average for New York City but lower than the national averages. With a few exceptions, the staff took a traditional approach to teaching through whole class lessons, textbook reading and worksheet drill. Under normal circumstances, the school would have been a likely customer for an integrated learning system. However, in this case, the school's computer teacher, who had a different vision, was able to play a leadership role and make use of the technology provided by the project (Bruder, 1990).

An AppleTalk network of approximately 50 Apple II and Macintosh computers includes a file server that allows for central storage of data, text, and programs. The environment contains a variety of word processing and database, desktop publishing and other tool software, including electronic mail which connects students and teachers both over the local and wide area networks (Newman, 1990b). Along with the technology, we introduced a year-long earth science curriculum designed in collaboration with the teachers (Brienne & Goldman, 1989).

Databases are used extensively both within and outside the earth science curriculum. During the lunch hour, students are found inventing databases of their favorite action figures. In social studies, students research almanacs and other sources to fill in databases about countries of the world, and figures from African-American history. In earth science, they examine databases of dinosaur fossils and earthquakes and create databases of the weather readings and indicators of seasonal change that small groups of students collect over a period of several months.

The primary means for supporting project work in Earth Lab is what we call "workspaces" which are folders on the file server and in which the work of the project, in the form of text, database, graphics and code files is stored. These workspaces, available to any computer on the school LAN, give groups a location for their work together. Students and teachers can be assigned to any number of workspaces. For example, workspaces are set up for pairs of students to work on writing assignments together. Other workspaces serve school-wide clubs or other projects. Each individual also has a personal workspace. The science work groups gave themselves names that were used for group workspaces on the network. Students share different data with different students or groups in the school, for instance, a science group, a noon hour club, and the whole class.

We find that with this approach to the use of a school LAN, teachers are better able to collaborate, students are better able to carry their work from one context to another, and the computer lab was increasingly used in a heterogeneous manner with several projects or groups from different classes working simultaneously. The following examples taken from our observations at the school illustrate these changes within the same categories used to describe the integrated learning systems.

Location

The school's computer network includes two separate labs, a satellite lab in a small room off one of the classrooms and network connections in several other classrooms into which computers can be moved as needed. The way the project workspaces were set up for groups and individuals helped to develop a sense of continuity not possible with an ILS.

For example, several students from different classes and different grades were editors for the school newspaper. The newspaper had a workspace on the network that students used for storing articles and other material for the newspaper. Beyond the editorial group, many students around the school contributed articles to the newspaper by sending them as messages through the electronic mail system to the editors. The common workspace made it easy for the editorial group to work at different times and places on the newspaper. The network makes the boundaries between classrooms and class periods more permeable. The ease with which any student could contribute to the newspaper and the identity of the group task that was supported by the workspace widened participation. Students became familiar with the network's function as a data organizer so that when other school projects such as editing a video newscast were started, students thought it quite sensible to create a workspace for their scripts, plans and edit lists.

Although the computer labs, housed in adjacent rooms, have enough computers to accommodate a class with a 1 to 1 ratio, we seldom see the computers used that way. Usually students work in pairs or small groups. Since there are spare computers, students from other classrooms which do not have their own computers and teachers on their prep periods also come to the lab to work on various projects. As a result, activities in the labs are very heterogeneous.

The Earth Lab interface, which displays for students and teachers lists of their project workspaces, enables them to work on any of their projects whenever they have the opportunity and inclination at any of the networked machines available to them. With this greater continuity over space and time, students can take greater initiative in following through with work on a project.

Curriculum and Assessment

The earth science curriculum developed for the initial field test, and the curriculum materials that the teachers have continued to develop over subsequent years have been interdisciplinary. As they worked on weather and seasonal change, students made connections to physics, math, writing, and social studies. The network system made classroom projects easier to manage and promoted collaboration among the teachers.

For example, at the beginning of the first year of operation some teachers in this essentially traditional school had doubts about the students' capabilities for handling the autonomy involved in small group project work. Having the small group workspaces on the network helped communicate to the teacher community that students were expected to do collaborative work. Where interdisciplinary projects become a more common feature of the curriculum, the workspaces give the students a clearer group identity or sense of project continuity and thus help in the classroom management. Instead of greater centralized control of individualized instruction, as is common in integrated learning systems, control can be distributed to the students.

The Earth Lab network made no attempt to provide a technological solution to the problem of assessing student progress or grading student projects which is the central function of integrated learning systems afforded by the hierarchical nature of the courseware. We have, however, begun exploring the use of group and individual workspaces as portfolios of student work. The notion of a portfolio is receiving growing attention among educators as an alternative means of assessment in which the stages of work on a project collected in a portfolio can provide insight to both the teacher and the student about the state of their work, and in retrospect about the process of learning (Gardner, in press). The workspaces currently serve as archives of the group or individual project work and so can serve the function of a portfolio in this sense. Assessment systems such as TextBrowser (Kurland, 1991) which serve a database and analysis function for the teacher's own evaluations of the students work (their writing in the case of TextBrowser) provide a division of labor between teacher and technology that is appropriate for project based school work.

Timeframe

The project workspaces provide continuity over time as well as location. Projects involving collecting weather data and data on seasonal change extended over many months. In some cases, projects may extend over years as new cohorts of students move through the school curriculum.

For example, one science group was analyzing their weather data trying to support a theory suggested by their impression that the last winter had been much milder than the previous one. They were comparing their data with that collected by last year's sixth grade class. When their theory was not supported by averages in the report generated with the database program, one of the students checked the data entered by their classmates. Several temperatures seemed unrealistic, e.g., several January days with highs of zero degrees. Suspecting an error attributable to missing values, they sent electronic messages to representatives of several other classes who had kept similar records and obtained actual data for the days in question.

The use of tool software requires a greater initial investment in order to bring students up to speed with the technology than is required for integrated learning systems which present small tasks and simple interactions with the technology. However, the availability of the Earth Lab system to students over a period of years and the consistency of the available tools has made it increasingly easy for teachers to introduce long-term projects as part of their curriculum. In the first year of operation, the sixth grade class spent several months on fairly simple introductory projects designed to familiarize them with the word processing, database and communication tools. Several years later, teachers are able to start immediately with substantial projects.

Conclusion

This paper has contrasted two very different uses of local area network technology in schools attempting to trace their relationship to the way the school organizes instruction, the way teachers work together and the opportunities students have for engaging in long term, open-ended projects. The relationship between technology and the organization of instruction is complex. An integrated learning system certainly does not cause schools to have a compartmentalized curriculum sequence and division of labor among teachers. It may, however, provide strong support for those tendencies and place barriers to teachers who may wish to change their mode of instruction. At the same time the use of these systems can be modified quite radically by, for example, distributing the computers among the classrooms where they can become more integrated with the ongoing work of the classroom. In the same way, a system like Earth Lab does not cause collaborative, interdisciplinary project work to happen. But for schools that are moving in that direction, it can provide some useful tools and mechanisms. Research and development on the organizational supports for school work is continuing in two critical areas: developing tools for students to take a larger role in the management of their own work and developing mechanisms for students and teachers to assess and reflect on the projects.

Preparation of this paper was supported by the Center for Technology in Education under grant number 1-135562167-A1 from the Office of Educational Research and Improvement, U.S. Department of Education to Bank Street College of Education. The Earth Lab project was supported by the National Science Foundation under grant number MDR 8550449 and by Apple Computer Inc. External Research.

Acknowledgements: For their comments on earlier drafts, I am grateful to Paul Reese, Jan Ellis, and Jan Hawkins.

References

Brienne, D., & Goldman, S.V. (1989, April) Networking: How it has enhanced science classes in New York schools... and how it can enhance classes in your school too. *Classroom Computer Learning*, 44-53

Bruder, I. (1990) Facing the future. *Electronic Learning*, September, 34-40.

Carnegie Forum on Education and the Economy (1986) *A nation prepared: Teachers for the 21st Century*. New York: Carnegie Corporation of New York.

Gardner, H. (in press) Assessment in context: The alternative to standardized testing. In B.R. Gifford and M.C. O'Connor (Eds.). *Future assessments: Changing views of aptitude, achievement and instruction*. Boston: Kluwer Academic Publishers.

Kurland, D.M. (1991) TextBrowser: A computer-based tool for managing, analyzing, and assessing student writing portfolios. Paper presented at the annual meetings of the American Educational Research Association, Chicago.

Lave, J. & Wenger, E. (1990) Situated learning: Legitimate peripheral participation. Palo Alto: Institute for Research on Learning.

Mageau, T. (1990) ILS: Its new role in schools. *Electronic Learning*, 10(1), 22-32.

Newman, D. (1990a) Opportunities for research on the organizational impact of school computers. *Educational Researcher,* 19(3), 8-13.

Newman, D. (1990b) Cognitive and technical issues in the design of educational computer networking. In L. Harasim (Ed.). *Online education: Perspectives on a new medium*. New York: Praeger Publishers.

Newman, D. (in press) Interpreting an intelligent tutor's algorithmic task: A role for apprenticeship as a model for instructional design. *A.I. & Society*.

Newman, D., Goldman, S.V., Brienne, D., Jackson, I., & Magzamen, S. (1989) Peer collaboration in computer-mediated science investigations. *Journal of Educational Computing Research*, 5(2), 151-166.

Schofield, J., Evans-Rhodes, D. & Huber, B.R. (1990) Artificial intelligence in the classroom: The impact of a computer-based tutor on teachers and students. *Social Science Computer Review*, 8(1).

Sherry, M. (Ed.) (1990) *The integrated instructional systems report*. Water Mill, NY: EPIE Institute.

Sizer, T.R. (1984) *Horace's compromise.* Boston: Houghton-Mifflin.

Radical Strategy Variability:
A Challenge to Models of Procedural Learning

Stellan Ohlsson and Nancy Bee
Learning Research and Development Center
University of Pittsburgh, Pittsburgh, PA 15260, U. S. A.

Abstract: Current theories of procedural learning typically assume that a learner has a single strategy for each cognitive task. This *single strategy assumption* is contradicted by data which show that each learner entertains a space of strategies for the task he or she is trying to master. We propose three quantitative regularities of strategy spaces. Current theories of procedural learning do not suffice to explain these regularities.

The Single Strategy Assumption

Current theories of procedural learning typically represent cognitive skills as sets of rules, and they analyze procedural learning as a sequence of revisions or extensions of an initial rule set. The mechanisms by which rules are revised or created (chunking, composition, constraint satisfaction, discrimination, generalization, and so on) differ from theory to theory, but the different theories share the assumption that a person has a *single* strategy for each task. At each point along the path to mastery, the learner has one and only one strategy for the task he[1] is trying to master. This *single strategy assumption* is implicit in the simulation models constructed in our laboratory (Ohlsson, 1987a; 1987b; Ohlsson & Rees, 1991), in the ACT* model (Anderson, 1989), in the Soar model (Newell, 1990), and in other models of procedural learning. The assumption is implicit but crucial, because the learning mechanisms in the various models are designed to operate in the context of successive revisions of a single rule set.

Theories which adopt the single strategy assumption explain variations over time as a result of learning—i.e., as *change* in a single strategy—and variations between individuals as a result of different learning histories and/or different initial strategies. However, data from several domains indicate that strategy variability goes beyond longitudinal change and interindividual differences. Learners construct not one, but an entire space of strategies for the task they are trying to master, and they choose a strategy from that space on a problem by problem basis. This view has been advocated for some time by Robert Siegler (see, e.g., Siegler and Robertson, 1982, pp. 287-299). We refer to intraindividual strategy variability at a moment in time as *radical strategy variability*, to distinguish it from longitudinal change and from interindividual differences.

The study reported here demonstrates radical strategy variability in the domain of fractions and suggests three quantitative regularities of strategy spaces. These regularities belong with the negatively accelerated learning curve as touchstones for theories of procedural learning.

Radical Strategy Variability in Fractions

Method. Seven fifth-grade students participated in 13 one-on-one tutoring sessions over a period of seven to ten weeks. The sessions lasted 20-30 minutes each. The subjects studied fractions with the help of a computerized learning environment called the Fractions Tutor (Ohlsson, Bee, & Zeller, 1989). The subjects were taught the initial segment of the fractions curriculum: the meaning of the fractions symbol, the concept of equivalent fractions, the skill of constructing equivalent fractions, and the skill of adding proper fractions. Their knowledge of these topics was assessed with a paper and pencil, off-line test that was administered both before and after the tutoring sessions. The subjects were asked to explain their answers to the test problems. Their explanations were tape recorded and transcribed verbatim.

We will focus on data from three types of test problems: (a) *comparison problems*, in which the subject was given two fractions and asked which is greater or if they are equivalent, (b) *equivalence problems*, in which the subject was given a fraction n_1/d_1 and asked to construct an equivalent fraction n_2/d_2 with a given denominator d_2, e.g., 12/15 = ?/5, and (c) *addition problems*, in which the subject was given two fractions and asked to compute their sum. Each subject solved twelve comparison problems, four equivalent problems, and four addition problems on each test occasion.

Results. The data have been analyzed from a pedagogical point of view in Ohlsson, Bee, and Zeller (1989). For present purposes, we will focus on strategy variability. The subjects' explanations were used to identify the strategies they used on each problem. Table 1 contains four examples of explanations and the strategies we inferred from them.

Table 1
Examples of Observed Strategies for Equivalence Problems

Problem	Subject's answer	Subject's explanation	Inferred strategy
1/5=?/10	1	"Top numbers have to be the same. Bottom numbers doesn't."	$n_2 = n_1$
12/15=?/5	2	"Because I subtracted ten from this to get five, and I subtracted ten from twelve and got two."	If $d_2=d_1-10$, then $n_2=n_1-10$.
12/15=?/5	3	"Three times five is fifteen."	$n_2*d_2=d_1$
4/8=?/2	12	"I added them [points to 4 and 8] together."	$n_2=n_1+d_1$

Table 2
Number of Strategies, Overlap, and Turnover Rate

Problem type	Pretest	Posttest	Overlap[a]	Turnover rate
Comparison problems	21	22	11	.50
Equivalence problems	13	11	2	.80
Addition problems	4	3	1	.67
All problems	38	36	14	.61

[a] Strategies observed both on the pretest and on the posttest.

Size of strategy space. The total number of strategies observed over all three problem types in the pretest was 38. See Table 2. The number of strategies per subject varied between five and 13 with an average of 8.3, confirming the that the subjects had more than one strategy for each of type of task. There was little change in these numbers between test occasions. We observed 36 strategies all in all on the posttest. The number of strategies per subject varied between five and nine, with an average of 7.0.

Oscillation rate. Table 3 shows the *oscillation rate*, defined as the proportion of problems on which the subject used a different strategy than the one he used for the preceding problem. The average oscillation rate was .45, i.e., the subjects switched strategy on approximately half the problems. There was no average difference between pretest and posttest. The oscillation rate on the pretest varied across subjects from .27 for Subject 4 to .73 for Subject 2. The range was similar on the posttest.

Table 3
Oscillation Rates for Each Subject and Test Occasion

Subject	Pretest	Posttest	Both tests
1	.40	.53	.47
2	.73	.53	.63
3	.47	.60	.53
4	.27	.33	.30
5	.27	.27	.27
6	.40	.27	.33
7	.60	.67	.63
M	.45	.46	.45

Usage rates. The fact that subjects have many different strategies and that they switched between them on a problem by problem basis does not tell us how frequently any one strategy was used. All strategies in a subject's strategy space might be used with equal frequency. However, this is not the case. The data show that the *usage rates* are highly skewed. Many strategies are rarely used and only a handful are used often. Figure 1 shows the distributions of usage rates for the pretest. Even though the turnover rate was over 60% (see below), the distribution for the posttest was very similar.

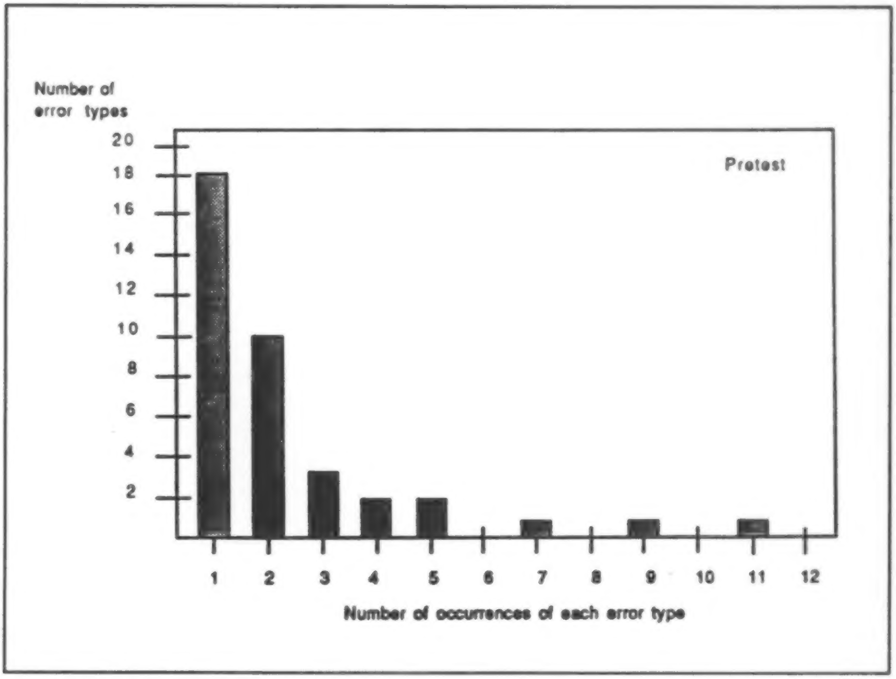


Figure 1. Distribution of usage rates for incorrect fraction strategies

An additional fact about usage rates is that the two most frequently used strategies on the pretest–to compare fractions by comparing their denominators (the larger the denominator, the larger the fraction), and the so-called freshman error (the sum of n_1/d_1 and n_2/d_2 is computed as $[n_1 + n_2]/[d_1 + d_2]$)–were also the most frequently used strategies on the posttest.

Turnover.[2] The fact that there were 38 different strategies on the pretest and 36 on the posttest invites the inference that there were only a few changes in the strategy space from pretest to posttest. However, this is not the case. As the fourth column ("Overlap") in Table 2 shows, only 14 strategies were observed on both pretest and posttest, so 24 strategies were dropped from the strategy space and 22 strategies were added during the seven to ten weeks of the study. The *turnover rate,* defined as the proportion of previously unobserved strategies on the posttest, is .61 on the average but varied between .50 and .80 across problem types (see Table 2, column five).

Summary and Discussion

The single strategy assumption is not correct. The learner does not make successive changes in a single strategy; instead, he constructs a space of alternative strategies for the task he is trying to master. In the present study, learners switched between seven to eight different strategies with an *oscillation rate* of approximately .50, i.e., they switched to a different strategy on every other problem. Large strategy spaces and high oscillation rates define the phenomenon of radical strategy variability. In the present study, this phenomenon was further characterized by three quantitative regularities: (a) The *usage rates* were highly skewed with only a handful of frequently used strategies. (b) Strategies were replaced with a *turnover rate* of 60% over seven to ten weeks. (c) Finally, the data suggest, but do not prove, that there is a negative correlation between high usage rate and turnover, so that frequently used strategies stay in the strategy space for longer periods of time.

The generality of these findings can only be partially documented in this short paper. High oscillation rates have been observed by Siegler in a series of studies of children's strategy choices on simple addition and subtraction problems (Siegler, 1987, 1989). Also, VanLehn (1982, pp. 10-11) reported intra-test strategy changes ("bug tinkering") in subtraction. Skewed usage rates have been observed by Nwana and Coxhead (1988) in a study of complex fraction problems, by VanLehn (1982, Appendices 2 and 3) in subtraction, and by Payne and Squibb (1990) in algebra. VanLehn also observed the relation between usage rate and turnover (or stability) that we observed here; the three most frequently used buggy substraction strategies in his study were also the most stable (VanLehn, 1982, p. 44). Radical strategy variability is not confined to mathematical tasks. In previous work we observed problem by problem strategy shifts in a verbal reasoning task (Ohlsson, 1984), and Simon and Reed (1976) observed intra-trial strategy shifts in a puzzle task.

Radical strategy variability implies that procedural learning cannot be conceptualized as a succession of revisions of a single rule set, but must be seen as the evolution of a strategy space. Accommodating this shift in view within current theory is likely to require the invention of new learning mechanisms. Furthermore, theories of procedural learning should explain the highly skewed usage rates, the high turnover rate, and the negative correlation between frequent usage and turnover (if the latter can be replicated). These regularities may pose as great a challenge for the theory of procedural learning as the negatively accelerated learning curve.

Notes

1. For the sake of brevity, we use "he," "his," "himself," etc., generically to refer to both genders.
2. Turnover is frequently discussed in the literature in terms of its opposite, i.e., the *stability* of strategies over time.

Acknowledgement: The preparation of this manuscript was supported in part by grant no MDR-8470339 from the National Science Foundation and in part by grant no N00014-89-J-1681 from the Cognitive Science Program of the Office of Naval Research.

References

Anderson, J. R. (1989). A theory of the origins of human knowledge. *Artificial Intelligence, 40*, 313-351.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Nwana, H. S., & Coxhead, P. (1988). Towards an intelligent tutoring system for fractions. *Proceedings of the conference on Intelligent Tutoring Systems*, Montreal, June 1-3, 1988. Pp. 403-408. Montreal, Canada: Universite de Montreal.

Ohlsson, S. (1984). Induced strategy shifts in spatial reasoning. *Acta Psychologica, 57*, 47-67.

Ohlsson, S. (1987a). Transfer of training in procedural learning: A matter of conjectures and refutations? In L. Bolc, (Ed.), *Computational models of learning* (pp. 55-88). Berlin, Germany: Springer-Verlag.

Ohlsson, S. (1987b). Truth versus appropriateness. In D. Klahr, P. Langley, & R. Neches, (Eds.), *Production system models of learning and development* (pp. 287-327). Cambridge, MA: MIT Press.

Ohlsson, S., Bee, N., & Zeller, P. (1989). *Empirical evaluation of a computer-based learning environment for fractions* (Technical Report No. KUL-89-07). Pittsburgh, PA: University of Pittsburgh.

Ohlsson, S., & Rees, E. (1991). Adaptive search through constraint violations. *Journal of Experimental and Theoretical Artificial Intelligence, 3*, 33-42.

Payne, S. J., & Squibb, H. R. (1990). Algebra mal-rules and cognitive accounts of error. *Cognitive Science, 14*, 445-481.

Siegler, R. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology: General, 116*, 250-264.

Siegler, R. (1989). Hazards of mental chronometry: An example from children's subtraction. *Journal of Educational Psychology, 81*, 497-506.

Siegler, R., & Robinson, M. (1982). The development of numerical understandings. In H. W. Reese & L. P. Lipsitt, (Eds.), *Advances in child development and behavior. Vol 16.* (pp. 241-312). New York, NY: Academic Press.

Simon, H., & Reed, S. (1976). Modeling strategy shifts in a problem solving task. *Cognitive Psychology, 8*, 86-97.

VanLehn, K. (1982). Bugs are not enough: Empirical studies of bugs, impasses and repairs in procedural skills. *Journal of Mathematical Behavior, 3*, 3-71.

Learning by Teaching

Srinivas Palthepu
National Centre for Software Technology
Gulmohar Cross Road No. 9, Juhu, Bombay 400 049 India

Jim Greer and Gordon McCalla
ARIES Laboratory, Department of Computational Science
University of Saskatchewan, Saskatoon, Canada S7N 0W0

Abstract: In this paper we describe a system which supports learning by having the student teach the system. The system is appropriate in situations where the student already has learned about a fact-based domain and wants to refine his/her knowledge. It also helps a student understand meta-cognitive skills useful in reasoning about such a domain. The system is initially a *tabula rasa* and learns about the domain through a guided interaction with the student. The system is not ignorant, however: it does incorporate a semantic network reasoning engine that helps it to deduce implications of the knowledge provided by the student. It also contains a set of heuristics that help it determine how to interact with the student: when to question him/her, when to give him/her initiative, when to point out various conflicts or incomplete aspects of the student's knowledge as provided to the system. The system is domain-independent and in addition to being a useful teaching tool is also an excellent experimental vehicle for exploring issues in pedagogy.

Introduction

A common observation among educators is that a body of knowledge is more completely understood after having being taught to someone else. Knowledge seems to be reinforced and internalized when the learner is put in the place of a teacher. In fact, understanding is sometimes defined as the ability to explain an observed phenomenon to oneself. Attempts to encourage the meta-learning technique of self-explanation have been explored and results indicate that self-explanation is correlated with better learning (Pirolli & Bielczyc, 1989; Ferguson-Hessler & de Jong, 1990).

Explanation of concepts to others has advantages over self-explanation. When a person is asked to explain a concept, that person is forced to make their implicit reasoning explicit. Explanation will bring out possible flaws in arguments, since through attempts to justify the reasoning, the person must examine previously unconsidered (and possibly non-existent) justifications (VanLehn, 1985). If an explanation is flawed, then the process of having constructed the explanation will characterize the cause of the flawed explanation either as an error in logic (or argument) or as a wrong assumption. Thus, diagnosing the causes of flawed explanation (either self-diagnosis or diagnosis by an ITS) will become easier (VanLehn et al., 1990). This approach to instruction, termed "reciprocal teaching" (Palincsar and Brown, 1984), has proven successful in traditional educational practice

(Moore, 1988). Related work has also been pursued by (Paquette et al., 1990) which places the student in the role of knowledge engineer.

Asking the student to provide explanations is fundamental in Socratic and apprenticeship methods of instruction. In addition, asking the student to engage in reciprocal teaching is a useful means of reviewing and reinforcing previously acquired knowledge regardless of the initial method of instruction. In the tutoring system described in this paper we focus on this latter approach: students are asked to "teach" the system. From a pedagogical perspective this helps a student reinforce and refine previously acquired domain knowledge, and also helps the student understand some of the key mechanisms for reasoning in the domain. These meta-cognitive skills may usefully transfer to other domains requiring similar reasoning skills. From a system design perspective, the tutoring system is relieved of the requirement of having vast and subtle domain knowledge. In fact we claim that the system need have no *a priori* domain knowledge; the system need only be capable of making inferences about the domain and of guiding the interaction with the student according to various pedagogical heuristics. This allows a general, domain-independent tutoring system to be built, and also provides an interesting laboratory in which to explore various pedagogical issues.

System Overview

We have designed a system that will facilitate learning by teaching. The system is a powerful reasoning engine and a pedagogical expert; but it has no knowledge whatsoever of the domain. The student learns by teaching the system his or her partially understood domain knowledge. The system is able to affect the student's learning by using its reasoning abilities and its pedagogical knowledge to help direct the student towards a better understanding of the domain. Figure 1 shows an architecture and information flow within the system.

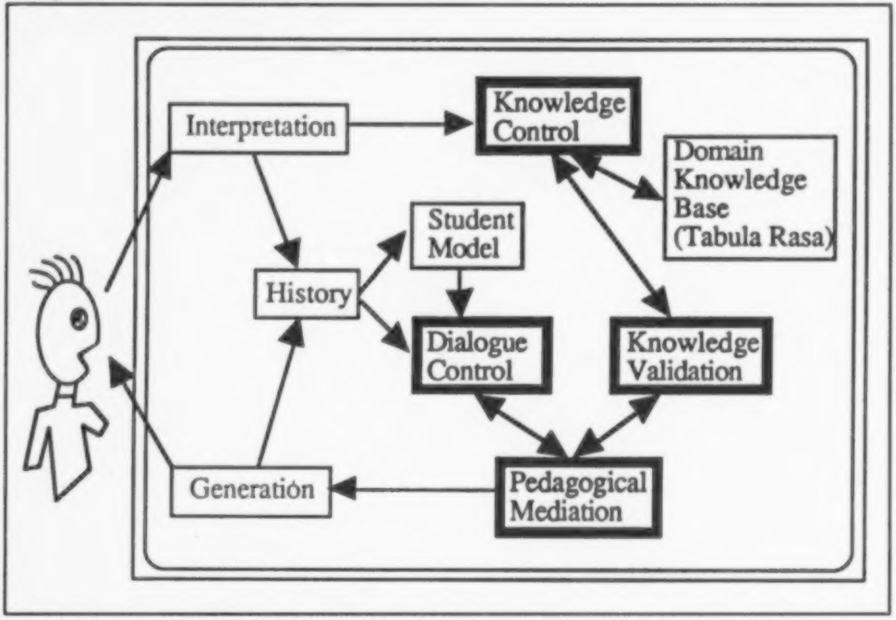


Figure 1. Learning by Teaching System Architecture

The system works in fairly straightforward steps: input from the student is interpreted; the domain knowledge base is updated; and a response is planned and generated. The Pedagogical Mediation component plans this response by drawing from two interacting knowledge sources, one concerned with Dialogue Control and the other concerned with Knowledge Validation. The bold outlined components are elaborated below.

The Knowledge Control component updates the Domain Knowledge Base with the information provided by the student. It also responds to queries from the rest of the system about the knowledge that is in the Domain Knowledge Base. In our system the Domain Knowledge Base is a standard semantic network. Thus, Knowledge Control can add, remove, or retrieve nodes, relationships between nodes, and properties of nodes. The choice of a semantic network knowledge representation scheme makes our system suitable for representing, storing, and retrieving factual knowledge. It should be noted that other inference and retrieval mechanisms would be appropriate for other types of domain knowledge.

Knowledge Validation is concerned with validating the knowledge provided by the student. The aim is to identify areas ripe for further exploration. There are various general types of Knowledge Validation heuristics: confirm existing knowledge, elaborate new knowledge, find contradictions, check for completeness of knowledge, etc. There are a number of specific flavours of these general types, for example, elaborating new nodes versus elaborating new properties of the nodes, or completing knowledge about the specializations of a node versus completing knowledge about the properties of a node. Furthermore, in a given situation each validation heuristic could apply at a number of different places in the knowledge base.

Dialogue Control is concerned with maintaining a natural ongoing interaction with the student. The parameters important to Dialogue Control include: providing continuity (e.g., focusing on a single section of the semantic network for some time); attempting to cooperate with student preferences (e.g., permit a focus shift if the student takes initiative), encouraging a shift in initiative (e.g., by providing a non-committal default response that encourages the student to take the initiative); and following "familiar" knowledge acquisition patterns (e.g., after elaborating new specializations, check whether they are a complete set). Dialogue Control uses the Student Model and History in making its recommendations.

Between them, Dialogue Control and Knowledge Validation may suggest many possible responses to the student. It is the task of Pedagogical Mediation to mediate among the many possibilities, ultimately choosing one response based on pedagogical preferences. For example, apparent contradictions discovered by Knowledge Validation would normally outrank all other considerations since it is important to clarify contradictions in the student's mind by pointing them out. In other situations, encouraging student initiative may suggest providing a non-commital default response rather than a more directed response aimed at elaborating specific parts of the knowledge base. In order to ensure efficiency, Pedagogical Mediation is not just re-active, but actively iterates between Dialogue Control and Knowledge Validation. For example, Pedagogical Mediation may decide that continuity constraints suggested by Dialogue Control enable Knowledge Validation to look only at parts of the semantic network relevant to the current dialogue focus.

Details of the System: A Sample Interaction

To give some idea of how the system could behave, the following sample interaction between student and system is presented (ST is the student, SM is the system). Note that the system is not yet fully developed, and the natural language dialogue in the following example is idealized.

ST: I will tell you something about the animal kingdom.
SM: OK. Please proceed.

In this case the student is initiating the tutoring session and the system provides a *default response* to encourage further input. Such default responses are frequently appropriate in situations where either the system does not understand the student or when it is desired that the student take the initiative in defining the next topic to be discussed.

ST: There are two types of living things. Plants and animals.
SM: So every living thing is either a plant or an animal.

The student's input causes Knowledge Control to augment the system's knowledge base by adding a node **living-thing** and two specializations of this node: **plant** and **animal**. The Knowledge Validation heuristic *check for exhaustive list* then is chosen by Pedagogical Mediation and the student is asked whether plants and animals exhaust the possibilities for living things.

ST: Yes. And all mammals are animals. And so are all reptiles.
SM: That means there are two kinds of animals?

The student confirms that plants and animals are in fact the only kinds of living things, so the knowledge base is appropriately modified to reflect this exhaustiveness. This modification will also block further firings of the *check for exhaustive list* heuristic, at least in regard to specializations of living things. However, the student also volunteers more information about animals, resulting in the nodes **mammal** and **reptile** being attached as specializations of **animal**. This re-establishes the *check for exhaustive list heuristic* as appropriate, this time to see if there are any other kinds of animals.

ST: No, actually there are more.
SM: Can you list them?

The student's response indicates that there are more animals than just mammals and reptiles. The heuristic *elaborate knowledge (specializations)* is chosen and suggests that the student be asked to list all of them. This heuristic could, of course, have applied to any non-elaborated node in the network, but is applied to the **animal** node in particular because Dialogue Control suggests the crucial need for dialogue continuity.

ST: Mammals
Reptiles
Insects
Birds
SM: Can you say something more about mammals?

After the student's response, two new specializations of animal are added (**insect** and **bird**), and the knowledge base is modified to indicate that the four specializations of animal are in fact an exhaustive list. Once again this blocks further firings of *check for exhaustive list*. The system now has a number of possible pedagogical options, including giving the student initiative by providing a *default response* or invoking the *elaborate knowledge (specializations)* heuristic on one of the four newly created specializations of animals. These heuristics, however, are overridden in this case by the *elaborate knowledge (properties)* heuristic that asks for properties of mammals. This heuristic is chosen for a number of reasons, including the need to get detailed student knowledge of the various concepts created so far and the desirability of having the student respond to a number of different kinds of requests. The heuristic is applied in particular to mammals since the student introduced mammals first, and thus may be more interested in mammals than other animals.

ST: Yes, all mammals have legs.
SM: OK.

The student supplies a property of mammals which is added to the knowledge base as one of the "slots" in the mammal node. Once again a *default response* becomes appropriate since it has been some time since the student has been given any initiative.

ST: Humans are mammals.
SM: Humans are animals?

A new node **human** is added as a specialization of **mammal** in the knowledge base as a result of the information provided by the student. The heuristic *confirm knowledge (transitive relationship)* asks the student to confirm the implicit relationship between humans and animals that is inferable transitively along "isa" links. Many possible rules could fire here; this one is chosen for variety; and the particular root of **human** is chosen to keep the dialogue focussed.

ST: Yes. Dolphins are also mammals.
SM: You mean dolphins also have legs.

The student's confirmation that humans are indeed animals does not result in any changes to the knowledge base, but does prevent further firing of the *confirm knowledge (transitive relationship)* heuristic rooted at **human**. The student provides more information, resulting in a new **dolphin** node as a specialization of **mammal**. The heuristic *confirm knowledge (inherited property)* is now chosen and the system asks the student to confirm that the inherited property is appropriate. Confirming inherited properties is a high priority of the system, since there are often exceptions that a student may not have thought of.

ST: No.
SM: But you said all mammals have legs and dolphins are mammals.

The student's negative response results in a property being attached to the **dolphin** node to the effect that dolphins do not have legs. The heuristic *find contradiction (inherited property)* then reacts to the conflict, and the system points out the problem to the student. Pointing out a contradiction is among the most important pedagogical goals, so the various contradiction heuristics (direct and indirect) are always given high priority.

ST: That may be true in general but there are exceptions. Dolphins do not have legs.
SM: Do you still believe that the rule "mammals have legs" is good in general?

The student confirms that the "no legs" property is indeed an exception. This property is then explicitly marked as an exception in the knowledge base so that the *find contradiction* rule will now be blocked from further firing. However, the *confirm knowledge (existing property)* heuristic is now appropriate since a natural possibility after an exception has been noticed is that there is now some doubt about the general rule.

ST: I think so.
SM: Are there other mammals that do not have legs?

The legs property attached to mammals is confirmed by the student. This then suggests the natural follow-up heuristic *elaborate knowledge (generalization)* to find out if there is a generalization of the **dolphin** node.

ST: Yes. Whales and seals.
SM: OK. Go on . . .

The student's affirmative answer suggests that there is a node **legless-mammal** between **mammal** and **dolphin**. This node is created and the "no legs" property is re-attached from **dolphin** to this new node. Two other new nodes, **whale** and **seal**, are created as specializations of **legless-mammal**, in addition to the existing **dolphin** node. The dialogue continues, but we will leave it now to consider some of the implications of learning by teaching.

Issues and Conclusions

The learning by teaching system described here is essentially a *tabula rasa* that has no prior knowledge about the domain but has the representational and inference capabilities to acquire new knowledge presented by the student. The knowledge in the system stems from the heuristics that direct the flow of dialogue and the selection of what to say when certain types of changes in domain knowledge occur. All of this means that the system, while focussed on fact-based domains, is independent of any particular such domain.

The system could be most usefully applied in situations where a student already "almost knows" the domain. In such situations the system would be a kind of study aid helping the student rehearse his/her facts, understand the direct and hidden relationships among these facts, discover where there are gaps in his/her knowledge, and most interestingly understand some of the important kinds of reasoning that can be done in fact-based domains. Such meta-cognitive skills may be the most long lasting residue left after extensive use of this system.

Our experiments to date have barely begun to explore the implications of the learning by teaching approach. We have only elaborated some of the acquisition heuristics, and these are limited to our semantic network knowledge representation scheme. An interesting direction to take this research would be to augment the knowledge base and inferencing capabilities of the system in order to expand on the number of domains that can be handled and to enhance the meta-cognitive reasoning skills to which the student is exposed.

We have also only just begun to investigate the various pedagogical options available to the system and how these tie into dialogue issues. Our current approach is quite *ad hoc* and more principled pedagogical and dialogue theories need to be developed. The interesting thing about our learning by teaching architecture, however, is that the roles of pedagogy, dialogue, and domain-based reasoning are clearly distinguished, making it relatively easy to experiment with a wide variety of different options in these areas.

So far in our experiments we have found that student history provides crucial knowledge to keep the dialogue focussed and pedagogically useful. The student model, however, seems less useful. An interesting open question is whether or not student modelling is critical in learning by teaching. A related question is whether our *tabula rasa* assumptions are justified. How useful would it be for a learning by teaching system to have domain knowledge, but disguise it from the student (at least most of the time)? This would allow the system to direct the student toward areas of interest that he/she has not explored, and would allow correction of student mistakes; however, it would also begin to muddy the learning by teaching paradigm and would destroy the system's intriguing domain-independence.

A final obvious future direction is to actually build natural language interpretation and generation modules, instead of faking it with "internal notation." This would allow the system to be experimentally validated with actual students. This direction has not been central to date since our entire effort has been on exploring various acquisition heuristics, dialogue heuristics, and pedagogical control options. The learning by teaching approach is a promising approach to providing student-centered, domain-independent, practical tutoring to students. That much of the approach still needs to be worked out doesn't detract from our excitement about its potential.

Acknowledgements: We would like to thank the Natural Sciences and Engineering Research Council of Canada for their financial support of this research, and the United Nations Development Program for funding mutual visits among the principal investigators.

References

Ferguson-Hessler, M. and de Jong, T. (1990). Studying physics tests: Differences in study processes between good and poor performers. *Cognition and Instruction*, 7(1): 41-54.

Moore, P. (1988). Reciprocal teaching and reading comprehension: A review*Journal of Research in Reading* 11(1): 3-14.

Palincsar, A.S. and Brown, A.L. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction*, 1(2): 117-175.

Paquette, G., Nadeau, R. & Longprè, M. (1990). A support system for knowledge engineering by the student. *Proceedings 8th Conference of the Canadian Society for Computational Studies of Intelligence:* 146-153.

Pirolli, P. and Bielaczyc, K. (1989). Empirical analyses of self-explanation and transfer in learning to program. *Proceedings Eleventh Conference of the Cognitive Science Society*, Hillsdale, NJ: Lawrence Erlbaum.

VanLehn, K. (1985). *Theory reform caused by an argumentation tool.* Tech Report ISL-11, Xerox PARC, Palo Alto, CA.

VanLehn, K., Ball, W., and Kowalski, B. (1990). Explanation-based learning of correctness: towards a model of self-explanation effect. *Proceedings Twelfth Conference of the Cognitive Science Society*, Cambridge, MA: Lawrence Erlbaum, 717-724.

Learning Biology Via Programming

Don Ploger
School of Education, 4533 Tolman Hall
University of California, Berkeley, CA 94720

Abstract: This research is organized around a central cognitive theme: students can learn about science by writing computer programs and by reflecting on the programs they have written. A program is an explicit and dynamic representation of relevant subject matter knowledge that can serve as the basis for extended, student-directed learning activities. At each stage of the learning process, students have representations that are meaningful to them, because they have written the programs. The representations can be extended and modified by further programming and debugging. This activity can be highly motivating and, at the same time, it requires students to be precise in their thinking.

This paper explores the space of student representational activity required by modern biology, including geometric representations of molecular structure, symbolic representations of genetic information, and the integration of diverse representations for topics in biology and medicine. I describe how a number of students have used this single computational medium to create, elaborate, modify, borrow and reconstruct programming representations for a wide range of biological topics. I then discuss the challenges of providing these educational opportunities to more students.

Students can learn about science by writing computer programs and by reflecting on the programs they have written. A program is an explicit and dynamic representation of relevant subject matter knowledge that can serve as the basis for extended, self-directed learning activities. At each stage of the learning process, students have representations that are meaningful, because they have written the programs. The representations can be extended and modified by further programming and debugging. This activity is highly motivating, and at the same time, it requires students to be precise in their thinking.

I take a constructivist view of learning: knowledge grows from prior knowledge and knowledge emerges in activity. Consequently, we are interested in how students think and on how they can learn to think better by reflecting on their own actions. My colleagues and I have developed a set of learning activities where students can investigate a range of topics. An essential component of any learning activity is the representation of knowledge. Programs have proved to be very useful representations for a number of students.

Although the theme of learning science via programming has been prominent in mathematics education (Papert, 1980; Abelson and diSessa, 1981) and physics education (diSessa, 1982), it has not been widely applied to biology education. In the physics of motion, the complex knowledge of the domain can be elegantly summarized by a few powerful mathematical equations, and it is possible to simulate motion with simple computer programs. In biology there is no small set of elegant equations to describe the domain. Therefore, learning biology via programming requires a different educational approach.

I will first describe the structure of knowledge in modern biology. I will then describe the computational medium used. Next, I will consider the space of representational activity with this medium. Finally, I will examine how two students learned about the same biological topic by creating distinctly different representations.

The Challenge of Learning Modern Biology

A wide range of representations have proved useful in modern biology, including geometric representations of molecular structure and symbolic representations of genetic information. In addition, many representations in medicine have been integrated in a coherent framework of modern biology.

I will examine a broad and important area of biology education: the relationship between biology and medicine. I will focus on an important area in biology and medicine—metabolic disease. To understand metabolic disease, students must integrate knowledge of molecular structure, cell physiology, and genetic information (Ploger, 1988). Students must be able to coordinate the relationship between normal and abnormal function. This depends upon a rich set of representations. This paper examines how students learn about phenylketonuria (PKU), a genetic disease that, without treatment, causes severe mental retardation in children. With treatment, children can live full and healthy lives. PKU can be understood clinically—a particular diet prevents the symptoms. PKU can also be described at the molecular level. Normally, the amino acid phenylalanine is metabolized to simpler compounds and ultimately removed. In PKU, the individual lacks an enzyme in the pathway for phenylalanine degradation, and phenylalanine builds up in the bloodstream and causes damage to brain cells. This example, therefore, illustrates two approaches to reasoning about biology and medicine.

A Flexible Programming Medium

To explore ways to learn this topic, I have selected Boxer, a computational medium intended to be used by people who are not computer specialists (diSessa and Abelson, 1986). It is possible to build problem representations simply by creating spatially-organized data objects. Because Boxer is a complete programming language, those representations can be modified by simple procedures that act on those data objects. Boxer can be a useful tool for students to learn about important issues of problem representation and problem solving strategy (diSessa, Abelson, and Ploger 1991). Students can learn the details of the operators simply by observing the effects of the procedures on the data representations. Then, students can organize those operators to form broader operations that are meaningful in the problem solving context (Ploger, in press-a; in press-b). The computational medium can help maintain the focus on reasoning, and students begin to understand biology without being overwhelmed by details.

I describe how a number of students have used a single computational medium to create, elaborate, modify, borrow and reconstruct programming representations for a wide range of biological topics. I then discuss the challenges of providing these educational opportunities to more students. It would be useful to have a medium where students could use to make and modify representations. It is especially useful if those representations are under computational control. This is valuable because students must make their representations explicit. Furthermore, they can see that there are a wide range of possible representations.

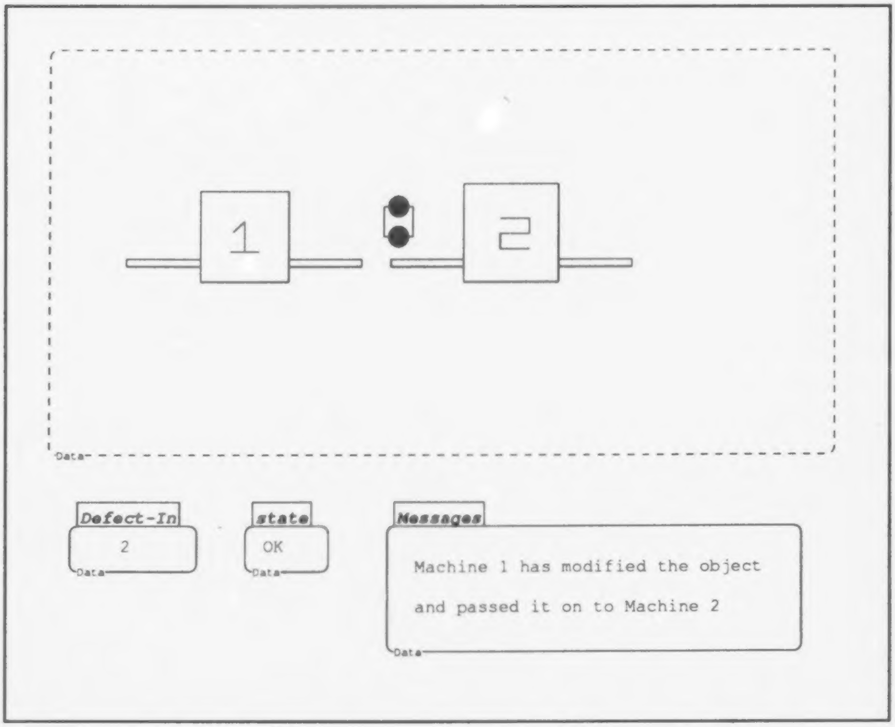
The Range of Representational Activity

Prior work has demonstrated that students can use Boxer to build representations for cellular and molecular processes (Ploger, in press-a), genetic information (Ploger, in press-b), and biology as abstract problem solving (Ploger, 1990). The present study is intended to illustrate the range of activities possible by describing two very different activities.

Representing an Analogy in Boxer

One way to represent knowledge of PKU is to create a simple analogy, an assembly line, to represent the abstract concept of a normal metabolic pathway. Furthermore, it is easy to demonstrate the consequences of abnormal function. If any machine in an assembly line is not functioning normally, there will be a decrease in product. Likewise, if any enzyme in a metabolic pathway is not functioning normally, there will be similar consequences.

One student, DM (age 17) learned about PKU and about broader issues in biology and medicine. His learning took place in a sequence of steps. First, he made a simple computational representation for an assembly line (Figure 1). The "product" on the assembly line is represented by a sprite (Boxer's version of the Logo turtle). The machines are drawn by Logo-type programs. With Boxer, however, it is also possible to represent the state of the system with visible variables.



Defect-In: 2

state: OK

Messages: Machine 1 has modified the object and passed it on to Machine 2

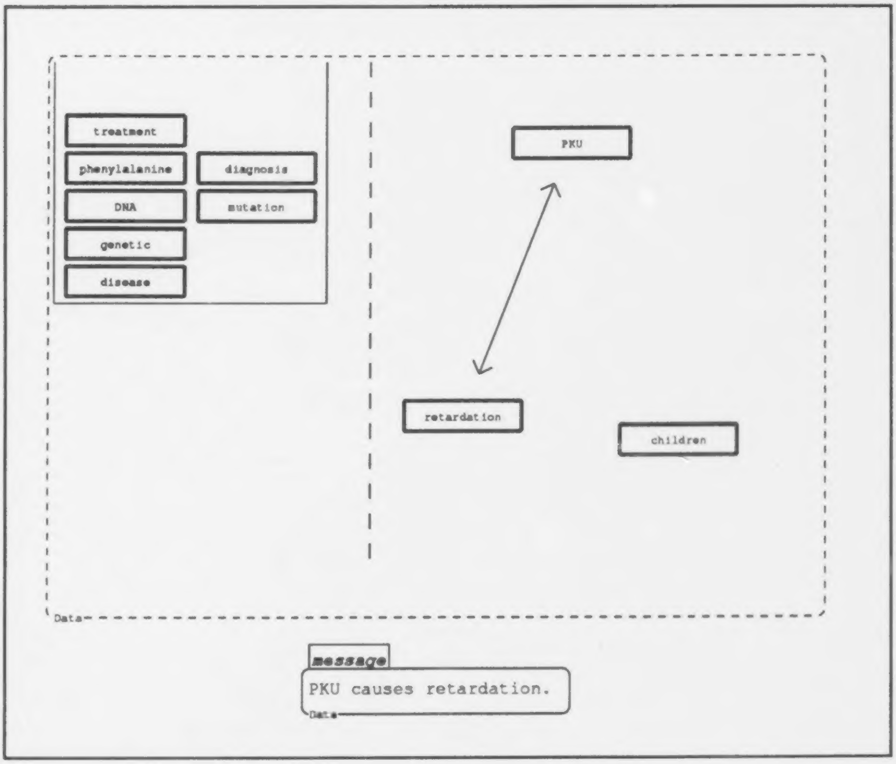
Figure 1. A program that simulates an assembly line

DM was able to write this program in a single two hour session. The result was a representation that he understood can could examine in detail. In a discussion with the investigator, DM realized that PKU is actually the inverse of an assembly line (the metabolic pathway breaks down phenylalanine). DM then built a dis-assembly line that recycled an object. As a result of this programming activity, DM had an organized set of representations. He began with very simple programs that could be easily understood, and then gradually created more complex programs that represented more subtle aspects of the biological topic.

An Explanation Construction Kit

Another student took a completely different approach to the topic. The challenge was to construct a visual representation for an explanation. DS (age 17) created a program where any concept could be represented by a sprite. In constructing an explanation, a person simply selects a term and moves it into the explanation space. The arrows are constructed by a program. The input to the program is a keystroke on each of the terms. The program draws an arrow between the two indicated terms and then inserts the appropriate text into the message box, based upon information the student entered into the system.

In Figure 2, there is a graphics box. On the left side of the graphics box is a bin of terms. On the right side of the graphics box are three terms: PKU, retardation, and children. There



```
PKU causes retardation.
```

Figure 2. The Explanation Construction Kit: A simple explanation

is an arrow between PKU and retardation. Below the graphics box, there is a data box named message. In this box, is the sentence: "PKU causes retardation."

This program was built with very simple Boxer commands. DS wrote simple procedures so that a line was drawn simply by moving the mouse to the appropriate sprite and pressing a button. The program made explicit the representation of knowledge about PKU. It also made it easy to describe PKU in simple terms.

DS was able to use the same program to give very different types of explanation, such as the biochemistry of the disease. This program uses the same aspects of Boxer data structure to provide a completely different structure of domain knowledge.

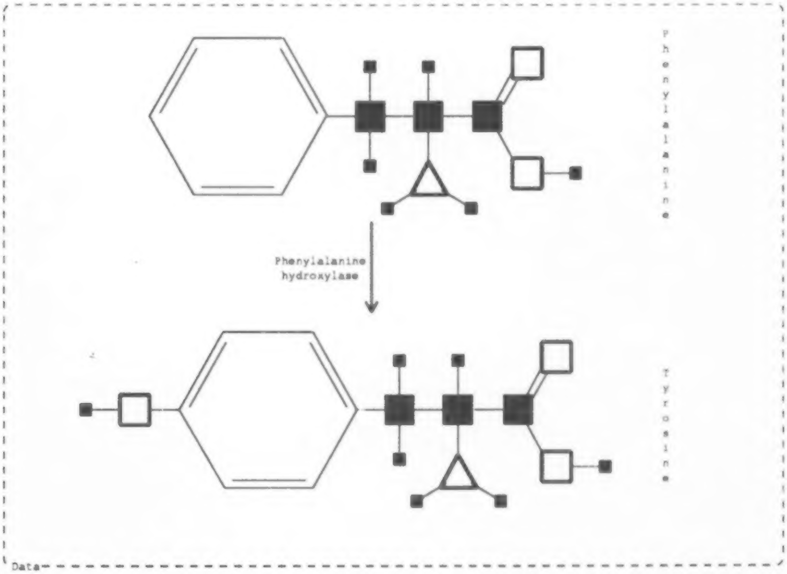


Figure 3. The Explanation Construction Kit: An explanation using representations of molecular structures

Figure 3 provides a very different representation, but it is easy to construct in Boxer. In this program, DS has drawn the crucially relevant enzymatic reaction. In healthy people, phenylalanine is converted to tyrosine. In persons afflicted with PKU, however, the enzyme necessary for this reaction is not functional. This program uses sprite graphics representations of molecular structures. These representations are computationally accessible and can be used to retrieve relevant information. In this activity, DS has easily created a computer notebook that not only stores information, but dynamically constructs a wide range of explanations that the student has constructed.

Discussion

These learning activities demonstrate the range of opportunities that students have to explore a complex topic by building computational representations. Projects included an extended analogical representation of important concepts and an explanation generator.

The explanation generator used simple terms as well as geometric representations for molecular structure. All of these representations were easily constructed by high school students in the same computational medium.

The programming medium makes possible extended student-directed learning. Students maintained ownership of–and interest in–projects over a long period of time. Students can readily create simple program representations. They can examine and extend those representations as their knowledge increases. The existence of external representations made a wide range of discussions possible–including the discussion of generating an explanation.

I have emphasized what students *actually* have done, not simply what they *could* do. When technology is widely available in schools, it seems likely that these activities could become readily available to a wide range of students. That technological advance is proceeding rapidly. It is important that–when the technology arrives–educational research can begin a serious investigation of modern biology.

This research is supported by a grant from the National Science Foundation to Andrea A. diSessa, MDR-88-50363, and by a contribution of equipment from Sun Microsystems, and a contribution to support students from Apple Corporation.

Acknowledgements: I thank Andy diSessa, Ed Lay, Dan McCormick, Dan Stoll, and Steve Adams for suggestions and contributions to this research. I thank Nina Gabelko and Lloyd Nebres of the Academic Talent Development Program at the University of California at Berkeley.

References

Abelson, H. and diSessa, A. (1981). *Turtle geometry*. Cambridge, MA: MIT Press.

diSessa, A. A. (1982) Unlearning Aristotelian physics: A study of knowledge-based learning. *Cognitive Science*, 6, 37-75.

diSessa, A. and Abelson, H. (1986) Boxer: A reconstructible computational medium. *Communications of the ACM*, 29, 859 - 868.

diSessa, A. A., Abelson, H., and Ploger, D. (1991) An Overview of Boxer. *Journal of Mathematical Behavior*. 10(1) (to appear)

Papert, S. (1980) *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.

Ploger, D. (1988). Reasoning and the structure of knowledge in biochemistry. *Instructional Science*. 17, 57-76.

Ploger, D. (1990). Problem solving with a flexible computer system: Reconstructing a discovery in biology. *American Educational Reasearch Association Special Interest Group for Artificial Intelligence and Education Newletter*. June 1990, p. 5-9.

Ploger, D. (in press-a). Reasoning and learning about mechanisms in biology. *Journal of Biological Education*.

Ploger, D. (in press-b) Knowledge Representation in Boxer: Learning about the genetic code via programming. *Journal of Mathematical Behavior*.

A Graphic Notation to Support Reflection in Scientific Reasoning

Kalyani Raghavan, Leona Schauble, and Robert Glaser
Learning Research and Development Center, University of Pittsburgh
Pittsburgh, Pennsylvania 15260, USA

Abstract: Our research on students' scientific reasoning with computer-based laboratories indicates the relationships between goal-driven, plan-oriented activity and success at discovering the laws and regularities. Students who learn the most are skillful at coordinating the complex set of subgoals that comprise experimentation. Students who fail to learn typically fail to reflect upon the inferential meaning of their experimentation activity. To support users in this reflective evaluation, we have developed a graphic Discovery and Reflection Notation (DARN) that depicts student activity with the laboratory from three perspectives: the Student View, the Plan View, and the Expert View. Each of the views organizes the student's activities in a different way. The Student View is a minimally edited, chronological trace of the student's experimental behavior. The second perspective, the Plan View, depicts changes in student plans for experimentation, and shows whether the activities subsequent to a plan are consistent with the plan. The Expert view, the 'expert's' eye view, organizes the data the student has generated with respect to its relevance to the laws students are attempting to discover. A current study is tracking the conversation of pairs of students working with DARN.

This paper describes the Discovery and Reflection Notation (DARN), a computer-generated graphic notation to support students' reflective evaluation of their self-guided experimentation in a computer-based laboratary. Our studies of undergraduates' scientific reasoning with computer-based laboratories that simulate phenomena in different subject-matter domains like microeconomics, dc circuits, and geometrical optics (Glaser, Schauble, Raghavan, and Zeitz, in press; Schauble, Glaser, Raghavan, and Reiner, in press; Shute, Glaser, and Raghavan, 1989) suggest a strong relationship between goal-driven, plan-oriented activity and success at discovering the laws and regularities underlying the domain. The most successful students are skillful at coordinating the complex set of subgoals that comprise experimentation, including generating experiments, predicting outcomes, interpreting results, recording and managing data, and developing and revising hypotheses. Students who fail to learn, in contrast, typically attend primarily to the rote procedures of operating the laboratories (for example, choosing variables, constructing experimental setups, and changing parameter values) and fail to reflect upon the inferential meaning of their experimentation activity. The purpose of the DARN is to encourage students to become more self-regulatory in carrying out these activities.

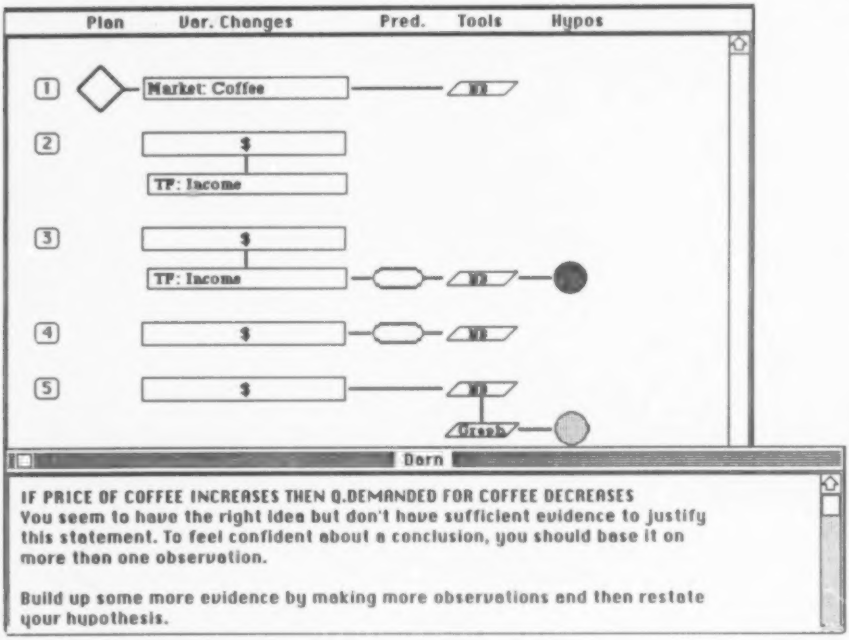
Reification as a tool for learning has been employed by Collins and Brown (1987) and Anderson, Boyle, and Yost (1985) in their intelligent tutors to teach problem-solving in algebra and geometry. They included inspectable problem-solving traces of student search in the form of tree graphs. However, the tree graph representation is not well suited to more complex forms of reasoning like experimentation, where there are many alternative ways to perform each component process and different paths to success in the overall enterprise. Our solution, inspired by Clancey's (1987) idea of treating the knowledge base as a reservoir of expertise to be used for different purposes, has been to devise a notation that graphically depicts student solution paths from different conceptual views, thus providing support for students to analytically reflect and debug their actions from different perspectives.

Our examples of the DARN will be shown in the context of Smithtown, the microeconomics laboratory.

When students work in one of the laboratories, a computer record of their various activities is maintained. DARN parses these records to organize them from three different viewpoints. Each of these viewpoints can be displayed in a graphical form in windows. Each view consists of *elements* arranged in a characteristic structural *pattern*. The three main views of student observations are: the Student View, the Plan View, and the Expert View. Each view embeds several layers of subviews that can provide progressively more specific information. At any time during a session, students may request one or more of the DARN views.

Student View. The Student View is a minimally-edited chronological trace of the student's actions with the computer laboratory. This view, a graphical summary of student actions over a period of time, closely resembles the student's own likely representation of his/her search path. The record of student actions is parsed and arranged in row-column fashion. The rows correspond to observational cycles and the columns are the five categories of actions: plans, variable changes, predictions, tools, and hypotheses. The purpose is to be able to review the classes of activities within and between observations at a glance to reveal the pattern of experimental behavior, for example, how frequently plans were changed, how many variables were changed at a time, regularity of predictions and tools use, and whether the search led to successful discovery of a principle or law.

Figure 1 depicts a Student View with its observation rows and action columns. The column titles appear at the top of the screen. For example, this student stated a plan in observation 1 and retained the plan throughout the rest of the observations. The next column corresponds to variable changes. The student made a change to the price per pound of coffee in observation 2 and also changed the average income. Students can optionally predict the expected outcome of their changes before observing the actual result. If a student choses to make a prediction, a prediction button is displayed in the student view, as in observation 3 in Figure 1. The laboratories provide three kinds of on-line tools, including a notebook to record data from an observation, a table-making facility to isolate collected data in order to sort and/or try out mathematical expressions to test perceived trends, and a graphing tool. In Figure 1, NB in observation 1 refers to the use of the notebook to record some information. Note that the student did not record the data corresponding to observation 2. The graph button in observation 5 refers to the use of the graph facility to plot selected points from the notebook. Students state their discovered law or regularity to the computer, which evaluates the hypothesis for accuracy and sufficiency of supporting evidence. This action is displayed under the column "Hypos." The button used to indicate this is a colored circle. If a student's statement is not definitive, either because of an inaccuracy in the statement or because of lack of supporting evidence, a red (medium gray) circle appears in this column. An accepted hypothesis is indicated either with a gold circle (light gray) (if the statement is correct for a specific case), or as a green circle (dark gray) (if the student has generalized evidence to two or more cases). In the example in Figure 1, the hypothesis stated at the end of observation 3 was unacceptable and the one at the end of observation 5 was acceptable



IF PRICE OF COFFEE INCREASES THEN Q.DEMANDED FOR COFFEE DECREASES
You seem to have the right idea but don't have sufficient evidence to justify this statement. To feel confident about a conclusion, you should base it on more than one observation.

Build up some more evidence by making more observations and then restate your hypothesis.

Figure 1. The DARN Student View

for the coffee market. Students can expand each of these buttons to review the stated hypothesis and the feedback given by the computer. For example, at the bottom of Figure 1 is the expanded view of the red hypothesis button attached to observation 3. In summary, this view portrays a student who started out with the intention of exploring the relationship between price and quantity demanded, changed multiple variables in observations 2 and 3, and reached a conclusion that was not definitive. Afterward, the student went on to a more systematic and successful investigation, changing price and observing the effect on quantity demanded. The purpose of the Student View, therefore, is to permit students to scan their sequence of experimental activity and to evaluate it for the basic issues of planfulness, systematicity, and persistence of search.

Plan View. The Plan View depicts whether a student is pursuing actions that are consistent with his/her stated plan. Planful experimentation was found to be a key factor in differentiating good and poor learners with the computer laboratories (Schauble, et al. in press; Shute, et al. 1989). When a student selects the plan view, the different plans stated by the student are displayed. When one of these plans is expanded, all observations made under this plan are parsed out of the student record and displayed. Figure 2 displays the view corresponding to a student's plan of exploring "price" and "quantity demanded." On the left are the observations made with respect to this plan. Colored vertical links indicate whether the observations generated are in fact relevant to this plan. In the figure, the solid green lines indicate that observations 3, 4, 5 are consistent with the plan, whereas the red dashed lines show that observations 1, 2 and 3 are not. Also displayed are the circles indicating conclusions, to depict if the student carried the plan to fruition. Each observation can be

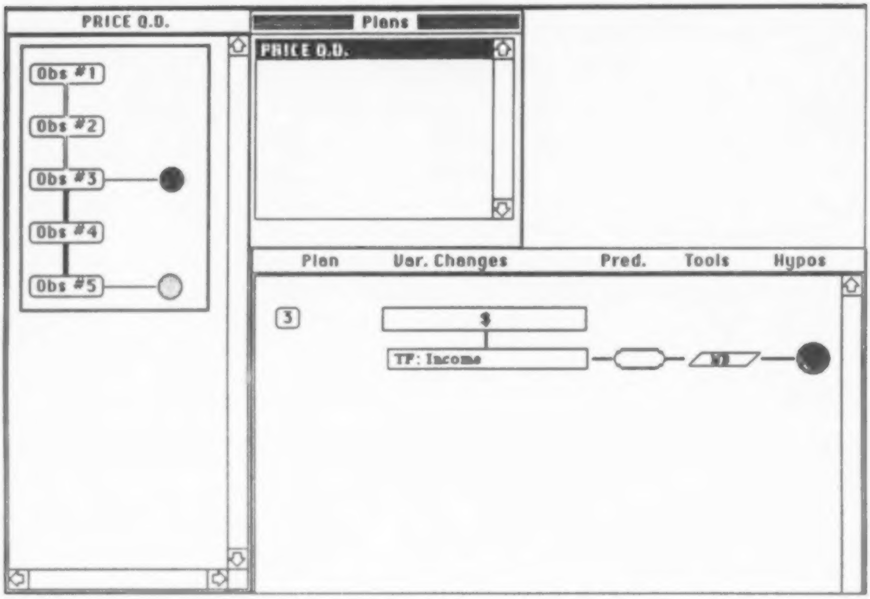


Figure 2. The DARN Plan View

expanded further to display the actions taken during that observation cycle. In Figure 2, the student has expanded Observation 3, and the experimentation activities that comprise this observation are seen on the right. The Plan View thus parses the ongoing flow of activity into units of search and permits students to evaluate (a) their persistence with a stated objective, and (b) how much of their search is relevant.

Expert View. This view organizes the data students generate with respect to the 'expert's' eye view based on an expert model of the domain. The student actions are parsed according to target concepts that the expert would have discovered with the same data. For example, in Figure 3, the Expert View displays three concepts, although the student has discovered only one of them. The observations in the box on the left constitute the evidence. The student may be unaware at this point that there are other relations that are discernible from the current set of observations. The purpose of the Expert View is to provide the impetus to go back and review the data to discover relations that have been overlooked. For example, students often fail to realize that different interpretations of the same data are possible.

To summarize, the Student View encourages the student to note important patterns in experimental behavior; the Plan View helps in assessing the coordination between plans and actions; the Expert View assists in judging the relevance of the observations to the concepts targeted in the laboratory.

A study is now being run to learn whether the DARN supports self-evaluative behavior in students working with the computer laboratories. Subjects are ten pairs of undergraduates who have not formally studied economics. Five pairs use Smithtown with DARN and the other five pairs, the control group, simply work with Smithtown. This study is tracking the conversation of the pairs. The dependent variables include the proportion of discussion related to reflection on experimentation as opposed to operational procedures or surface features of Smithtown.

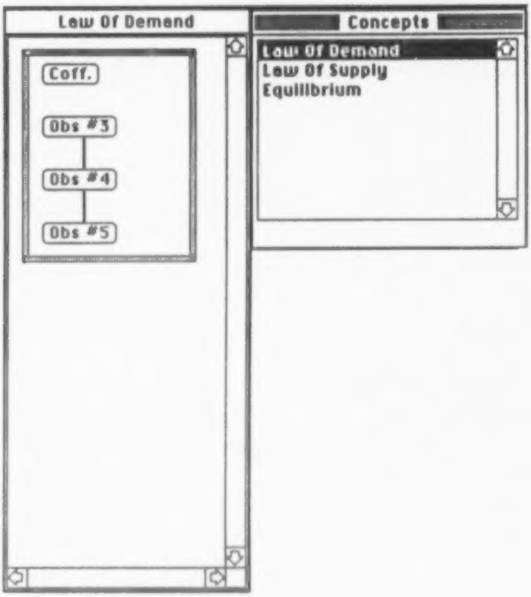


Figure 3. The DARN Expert View

Acknowledgement: Preparation of this article was sponsored in part by the Center for the Study of Learning (CSL) at the Learning Research and Development Center of the University of Pittsburgh. CSL is funded by the Office of Educational Research and Improvement of the U.S. Department of Education. Additional contributors to developing the DARN were Jamie Schultz, Colleen Zeitz, and Clark Quinn.

References

Anderson, J. R. Boyle, C. F. and Yost, G. (1985). The geometry tutor. *Proceedings of the International Joint Conference On Artificial Intelligence* : 1-5, Los Angeles, CA.

Clancey, W. J. (1987). Knowledge-Based Tutoring: *The GUIDON Program.* Cambridge, MA: The MIT Press.

Collins, A. and Brown, J. S. (1987). The computer as a tool for learning through reflection. In H. Mandl & A. M. Lesgold (Eds.). *Learning Issues for Intelligent Tutoring Systems.* New York: Springer-Verlag.

Glaser, R. Schauble, L. Raghavan, K. and Zeitz, C. (in press). Scientific reasoning across different domains. In E. DeCorte (Ed.), *Computer-based learning environments and problem solving.* New York: Springer-Verlag.

Schauble, L. Glaser, R. Raghavan, K. and Reiner, M. (in press). Causal models and experimentation strategies in scientific reasoning. *Journal of the Learning Sciences.*

Shute, V. J. Glaser, R. and Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In P. L. Ackerman, R. J. Sternberg, and R. Glaser (Eds.). *Learning and Individual Differences.* New York: W. H. Freeman.

Modeling Example Elaboration Strategies

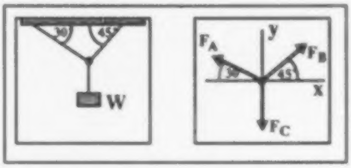
Peter Reimann and Thomas J. Schult
Department of Psychology
University of Freiburg, 7800 Freiburg, Fed. Rep. of Germany

Abstract: We outline a computer model that incorporates different strategies for acquiring problem solving knowledge from worked-out examples. Based on evidence from psychological research, an active, expectation-driven strategy for example processing is developed and contrasted with a passive, data-driven strategy. Both strategies deal with the initial phases of learning, exploiting episodic knowledge before relying on general knowledge. With the computer simulations, we want to find answers to the question of what kind of information must be retained from examples to make them useful for analogical problem solving. The active example processing strategy proofs to be superior to the passive one since it forces the learner to explicate the reasons for decisions left implicit in the example, thus leading to deeper understanding of the actions taken in the example. We believe our work lays the groundwork for building computerized tutorial systems that teach students how to learn better from examples.

The long term goal of the research reported here is to develop a computer program that helps students with learning from worked-out examples. The program receives the same example as a human student and tries to comprehend it, aided by domain-specific knowledge and an explicit strategy for example elaboration. The process and the outcome of the example understanding attempts form the basis for a dialogue between the machine and the human student. Both to generate output that is potentially meaningful for the student and to enable the program to follow the student through her elaboration activities, it needs information about example comprehension strategies as put to use by human learners. Therefore, a prerequisite for the above tutorial scenario are computational models of elaboration strategies and the knowledge they process. Work on this component is the focus of our current research activities. On the following pages, we first discuss the role examples play in problem solving and learning and then report on the development of computational example understanding models.

The Role of Examples in Problem Solving and Learning

Students tend to rely extensively on worked-out examples, even in domains where the solution knowledge is explicated (or even formalized) and the solution principle is known to them. For example, Chi, Bassok, Lewis, Reimann and Glaser (1989) observed that students rarely use a general problem solving procedure described in a physics textbook even after intensive study, but instead rely frequently on worked-out examples of the form shown in Figure 1 when solving mechanics problems (see also Chi and VanLehn 1991).



(1) The left figure shows an object of weight W hung by massless strings. (2) Consider the knot at the junction of the three strings to be "the body". (3) The body remains at rest under the action of the three forces shown in the right figure. (4) Suppose we are given the magnitude of one of the three forces. (5) How can we find the magnitude of the other forces? (6) F_A, F_B and F_C are all the forces acting on the body. (7) Since the body is unaccelerated, $F_A + F_B + F_C = 0$. (8) Choosing the x- and y-axes as shown, we can write this vector equation as three scalar equations: (9) $F_{Ax} + F_{Bx} = 0$, and (10) $F_{Ay} + F_{By} + F_{Cy} = 0$. (11) The third scalar equation for the z-axis is simply: (12) $F_{Az} = F_{Bz} = F_{Cz} = 0$ (...)

Figure 1. A worked-out example for a mechanics problem (Halliday and Resnick 1985)

However, the main problem with learning from examples is that students do not generalize over specific example solutions. That leads to the inability to transfer from the worked-out solution to problems that are not quite similar. In particular, it is often reported that students access examples primarily based on superficial similarities between the example and the problem at hand, and that they map solution steps from the example to the problem solution syntactically, leading to the inclusion of steps into a solution attempt that are either unnecessary or, worse, do not lead to a solution (Ross 1989).

An explanation for the conservative generalization strategy shown by many students (with the extreme of not generalizing at all) might be that examples do not contain the necessary information to perform sensible generalizations. In particular, examples do often not contain the reasons for *why* a certain step in the solution is performed. Consider the example solution provided in Figure 1. In statement 6, it says that the forces shown in the force diagram are "all the forces acting on the body." Yet, the example contains no procedure that describes how this could be determined by the student. In the absence of rationales for problem solving decisions, it is hard to decide what the essential features of an example are and what the superficial ones are, i.e., those one can generalize over.

How can we help students to better grasp the rationales for decisions made in worked-out examples? One solution would be to include this information explicitly in the examples themselves, enriching them with information on goal and operator selection. But, as Chi and Bassok (1989) mention, this strategy might not lead to success. The main problem is that this kind of information would not be tailored to the individual student, thus being redundant for one student, not comprehensible for another. Information that does not fit the individual learner is often ignored by the student or, worse, confuses her.

A better approach is to teach students to themselves become better in using information that is contained in examples. In the following, we will describe the components of a strategy for learning from worked-out examples that makes it possible to use them more intelligently for problem solving. In particular, the strategy leads to an example representation that allows to access the example based on structural features of the domain and to adapt the example solution in a flexible and correct manner.

Models of Example Elaboration Strategies

The general design of the strategy is based on insights into the process of text comprehension and understanding (e.g., Brown, Collins, and Harris 1987) as well as on research concerning the dynamic character of human memory and its influence on problem solving and understanding (Schank 1982; Kolodner and Simpson 1989). To account for the specific sort of text on which learning is based in our case: worked-out examples in mechanics, we further rely on observations made by Chi et al. (1989). There it was analyzed how students acquire problem solving knowledge concerning mechanics by studying worked-out examples. The study revealed important differences between successful and not so successful students, success measured in terms of correct solutions to problems. Successful students mentioned more often that they didn't understand a certain part of the worked-out example. Besides this difference in monitoring understanding of the example text, successful students also engaged in a series of activities to overcome their problems: They elaborated on the relations between a particular step in the example and the goals behind that step. They further attempted to come up with a specification of conditions that could explain why the operator under question was applied. Finally, they elaborated on the effects the application of an operator had beyond those mentioned in the example. The not so successful students displayed either none or considerably less of these elaborative inferences.

To capture the essential differences between successful and less successful learning from examples in form of computational models, we treat the process of example elaboration as a plan recognition task where the understanding system has to encode a given example solution in terms of problem solving goals, operators, and relations to domain concepts. Figure 2 shows the main components of the model. It comprises modules for example comprehension and problem solving. The problem solving component demonstrates the competence of the system. It attempts to solve new problems building on knowledge mainly acquired by learning from examples. Both the problem solving module and the example understanding module are hybrid systems, combining a case-based approach that uses episodic knowledge ("cases") about example solutions with general but weak inferencing methods.

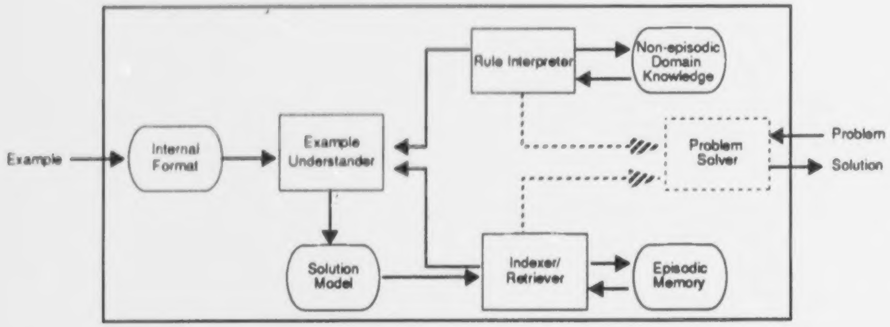


Figure 2. Model components

Most important for the current discussion is the example understander. This component takes a worked-out example as input (in a propositional format) and generates a model of the example solution that is then stored into Episodic Memory. To construct the example solution model, the system calls up episodic knowledge about former examples and non-episodic domain knowledge. The general control decision is to rely on episodic knowledge before attempting to use more abstract domain knowledge, this time for the example comprehension task. Two strategies are contrasted: an active, expectation-driven one that leads to a hierarchical representation of an example and allows for problem solving by derivational analogy (Carbonell 1986), versus a passive example processing strategy leading to a flat representation that enables the problem solver to use an example by transformations based on syntactical similarities.

The distinction between the two strategies as we make here is an idealization. In reality, the two strategies share more components than we assume here for purposes of exposition. We further suppose that the passive learning strategy consists of a subset of the active strategy's components. Therefore, we describe the active strategy in some detail and suggest only shortly which components are missing in the passive strategy.

Active Example Elaboration Strategy

Under this strategy, the system *monitors* its understanding of the example by checking for the steps in the example solution whether they cohere with its knowledge about problem solving in the domain. It tests its understanding in an active, expectation-driven manner: The system *predicts* the next example step and *compares* its prediction with the step actually appearing in the example. Problems in understanding are identified by false or missing predictions. Having identified an understanding problem the system tries to *explain* the offending part of the example, thus resorting to a cognitively more demanding mode of example processing. This includes attempting to derive the action(s) observed in the example from its episodic and general knowledge about the domain as well as to come up with alternate solution plans. In summary, this strategy reflects many aspects of the performance of successful subjects from the study of Chi et al. (1988) and incorporates general principles of text comprehension (e.g., Brown et al. 1978).

We are currently implementing this strategy in a system called AXE, the *A*ctive e*X*ample *E*laborator, using KEE and CommonLisp. AXE "reads" an example statement by statement and attempts to formulate expectations about the content of the next statement. A statement comprises propositions describing either (a) a goal selection, or (b) an operator selection (law, equation, inference rule), or (c) a problem solving state. Thus, AXE tries to predict (a) which goal will be worked on next, (b) which operator will be applied, or (c) what effects a goal or operator selection will have on the solution. These predictions, or expectations, are generated as follows.

Step 1 - Case-based expectation formation. After having read the first lines of an example where the problem is described in terms of objects and their relations, values given, and values desired, AXE performs a look-up in its Episodic Memory to find out whether it has encountered a similar example before. Episodic Memory has the structure of a MOP-based memory (Schank 1982) that indexes problem statements and solution steps as they occur in worked-out examples. Similarity at this stage is defined in terms of overlap in features of the problem statement. If AXE finds a similar example, it will be used to derive expectations for the new one. AXE retrieves the first operator from the old solution and uses it to predict the next state of the current example. Operators in Episodic Memory are indexed in terms of the goal(s) they serve, their necessary preconditions, effects, and further justifications. Justifications capture information not expressed in local goals and preconditions about the circumstances under which an operator can reasonably be applied (e.g., which other operators must be available, what the goal tree must look like) and in which circumstances the operator should not be applied. When transferring an operator from an old solution to the current example, AXE checks whether the operator can be applied (are the necessary conditions satisfied?) and whether it should be applied (are the justifications satisfied?). If not all of these conditions are met, the specific operator will not be applied. In this case, AXE will attempt to suggest at least the *class* of operators it expects to see in the next example statement. For instance, it may only be able to predict that the next solution step will deal with "rotating the reference frame" without being able to predict the details of that rotation.

Step 2 - Rule-based expectation formation. If no similar example is found, AXE attempts to formulate a prediction by using its non-episodic domain knowledge in a forward-reasoning mode. Non-episodic knowledge is represented in form of rules. Again, AXE can at times predict the class of operators instead of a specific operator since its knowledge about domain operators is organized in a rule hierarchy. After Step 1 or 2, AXE does either have an expectation about the next state of the example solution or has failed to derive one. Having no expectation is considered a failure and the system continues with Step 5, trying to repair this failure. If an expectation has been derived, it continues with Step 3.

Step 3 - Comparison. In the next step, the expectations about the example solution state are compared with the actual situation as described in the example statement. This comparison leads to an evaluation of the expectation. It may be (a) specific, (b) too general, or (c) insufficient, depending on whether a specific state can be predicted (a), or whether only a general state of affairs can be predicted (b), or whether the expectation covers only part of the effects resulting from the application of the operator (c). It is further determined whether the expectation is correct or wrong.

Step 4 - Continuation. In the case of a specific and correct hypothesis, AXE goes on and tries to formulate expectations for the next example state. Before that, the correctly predicted operator instantiation is stored into Episodic Memory in terms of its differences to the operator instantiation retrieved.

Step 5 - Case-based repair. In all the other cases (no expectation, wrong, insufficient or too general expectation) AXE tries to learn from the expectation failure. The failure-driven repair is performed first in a case-based manner, relying on episodic knowledge before attempting to use more general, but search-intensive repair strategies. AXE looks into Episodic Memory to see whether it has made a similar expectation failure before and/or has encountered a similar example state before. This information is used to update the example solution model. The now successfully explained solution step is stored into Episodic Memory and the system goes on predicting the next example statement.

Step 6 - Rule-based repair. If an expectation failure cannot be explained using Episodic Memory content, the system attempts to *reconstruct* the plan that might underlie the actions taken in the example part. This mode of example processing is particularly important during initial learning, since in this stage the learner possesses only a fragmented episodic knowledge structure and therefore cannot recognize an example as the "same old story". During plan reconstruction, the system resorts to its non-episodic domain knowledge and tries to find an operator or a sequence of operators that could have generated the problem solving state as displayed in the current example statement. If AXE finds a sequence that connects the statement in the example with a currently active problem solving goal, it stores the sequence into Episodic Memory. In effect, the system can from now on substitute matching for search when encountering similar example parts.

Step 7 - Copying the example solution step. If the plan reconstruction attempts fail, the system will store the specific representation of the example statement (an operator and its bindings) under the recent goal it was working on. Although AXE does not understand why this step was taken, it can at least copy the solution step when encountering a similar situation in a problem solving context.

An additional step - counterfactual reasoning. Whenever AXE succeeds either in predicting a part of the example solution (Step 1 or 2) or in recovering from an expectation failure (Step 5 or 6), it engages in a kind of counterfactual reasoning. After having established goal-action links by solution step recognition or reconstruction, starting from the goal it is asked: What other means exist to realize this goal? And why were they not chosen in the example solution? To come up with answers to these two questions, alternate solution paths are generated by working forward form the respective goal, applying domain rules. Doing that, AXE looks for problems with the alternate solution plans, problems that can be classified into categories such as *failed constraint*, *excessive costs*, *failed result*, or *bad side effect* (Collins 1987). The first step of the alternate sequence with the explanation of why it failed is stored in Episodic Memory. By learning from counterfactual reasoning, the system acquires decision heuristics, thus enriching the necessary conditions of operators with further justifications. Evidence for reasoning similar to this counterfactual inference component was found in the most successful student of the Chi et al. (1989) study.

Passive Comprehension Strategy

A word on the passive strategy that would reproduce aspects of poor learners' performance. It can be described in terms of differences to the active elaboration strategy. Missing monitoring of understanding is explained by assuming that the learner does not form expectations about the next parts of the example solution, but that she first reads the statement and then tries to explain: the understanding process is *data-driven*. Further, the kinds of explanations derived for a solution step are more local ones: Instead of trying to relate the observed solution step to higher-level goals and to constraints stemming from principles of the domain, the passive learner only tries to relate the step to the last step observed by subgoal-, effect- and enable-links. Finally, the passive strategy does not include any attempts to engage in counterfactual reasoning.

This active/passive distinction captures the end-points of what indeed is a continuum: "Real" students will hardly work so actively or so passively. We take these extremes as a starting point and will develop models that are psychologically more plausible by moving along the dimension spawned by these two models.

Effects on Problem Solving

Having implemented the central components of an active and a passive example processing strategy, we hope to be able to demonstrate in the next step that these differences in learning will lead to expected differences in analogical problem solving. That is, the active learner when working on a new problem should be able to retrieve from its Episodic Memory example solutions or parts of them based on features that go beyond literal similarity and should be able to adapt the example steps in a more flexible and correct manner than it is possible for the passive learner.

We then want to use these results for instructional purposes: To demonstrate to students the value of knowledge-based elaborations of worked-out examples and to give them a first model of *how* to elaborate on examples. Although the advantages of expectation-driven comprehension have been shown before for various sorts of texts (e.g., Brown et al. 1978), it has to our knowledge not be done for this specific, but important kind of text: worked-out examples. In addition, general theories of text comprehension usually presuppose a rich, well-structured knowledge base containing many generalizations, for example plans and scripts. Our model tries instead to capture how comprehension can function based on sparsely developed knowledge that is furthermore to a large degree not generalized, but episodic. This model should therefore help to support students in the initial phase of learning to solve problems, a phase in which many students experience more frustration than insight.

The research reported here is supported by a grant from the Deutsche Forschungsgemeinschaft (Re 814/1-1).

References

Brown, J.S., Collins, A., and Harris, G. (1978). Artificial intelligence and learning strategies. In: H. O'Neill (Ed.), *Learning Strategies*, pp. 107-139. New York: Academic Press.

Carbonell, J.G. (1986). Derivational analogy: A theory of reconstructive problem solving and expertise acquisition. In R.S. Michalski, J.G. Carbonell, and T.M. Mitchell (Eds.), *Machine Learning. An artificial intelligence approach, Vol. 2*, pp. 371-392. Los Altos, CA: Kaufmann.

Chi, M.T.H., Bassok, M., Lewis, M., Reimann, P., and Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. *Cognitive Science* 13:145-182.

Chi, M.T.H., and Bassok, M. (1989). Learning from examples via self-explanations. In L. Resnick (Ed.), *Knowing, learning, and instruction. Essays in honor of Robert Glaser*, pp. 251-282. Hillsdale, NJ: Erlbaum.

Chi, M.T.H., and VanLehn, K. (1991). The content of self-explanations. To appear in: *Journal of the Learning Sciences*.

Collins, G.C. (1987). *Plan creation: Using strategies as blueprints*. Ph.D. thesis, Yale University.

Halliday, D., and Resnick, R. (1985). *Fundamentals of Physics*. New York: John Wiley & Sons.

Kolodner, J.L., and Simpson, R. L. (1989). The MEDIATOR: Analysis of an early case-based problem solver. *Cognitive Science* 13: 507-549.

Ross, B.H. (1989). Distinguishing types of superficial similarities: Different effects on the access and use of earlier problems. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 15:456-468.

Schank, R.C. (1982). *Dynamic memory. A theory of reminding and learning in computers and people*. New York: Cambridge University Press.

GIL: Scaffolding Learning to Program with Reasoning-Congruent Representations

Brian J. Reiser, Ron Beekelaar, Alka Tyle, and Douglas Merrill
Cognitive Science Laboratory
Princeton University, Princeton, NJ 08542

Introduction

Much recent research on interactive learning environments investigates techniques for facilitating students' understanding as they tackle new knowledge domains. A key difficulty in many such domains is that the syntax of a problem's solution does not reflect the reasoning processes required to construct the solution (Anderson, Boyle, & Reiser, 1985; Collins & Brown, 1988). Our research on GIL, *Graphical Instruction in LISP*, investigates how to support students' learning and promote productive mental models as they learn to solve problems in the domain of computer programming (Reiser, Kimberg, Lovett, & Ranney, 1991; Reiser, Ranney, Lovett, & Kimberg, 1989). GIL is an intelligent tutoring system that responds to student errors and requests for guidance with explanatory feedback generated from its understanding of the problem and the student's solution so far. A major focus of this work examines how to embed the tutoring within a visual representation that provides a better scaffolding for students' reasoning than the traditional form of programs (Reiser, Friedmann, Gevins, Kimberg, Ranney, & Romero, 1988). GIL's representation was designed to be more congruent with the way novices reason about computer programs than the traditional text form of programs. In this paper we discuss how GIL is designed to facilitate students' reasoning about LISP programs, and present some preliminary comparisons of students learning to program using GIL with other students learning in a traditional programming environment.

Programming with Graphs in GIL

GIL uses a graphical programming interface designed to reflect more of the underlying structure of programs than their traditional text form. The goal of the design is to provide a closer fit with the reasoning steps students make, and thereby require less (potentially distracting) effort to reformulate a solution into the structure required for the program. GIL students build programs by connecting icons representing program constructs in a graph, rather than by defining LISP functions in their traditional text form. A completed program in GIL consists of a graph specifying a chain of LISP functions to transform input data into appropriate output. GIL graphs are designed to be more readily understood than the isomorphic text versions (cf. Larkin & Simon, 1987). The flow of control in this graph more explicit than in the text form of programs, and reasoning chains are represented by branches of a graph joined together to achieve a goal.

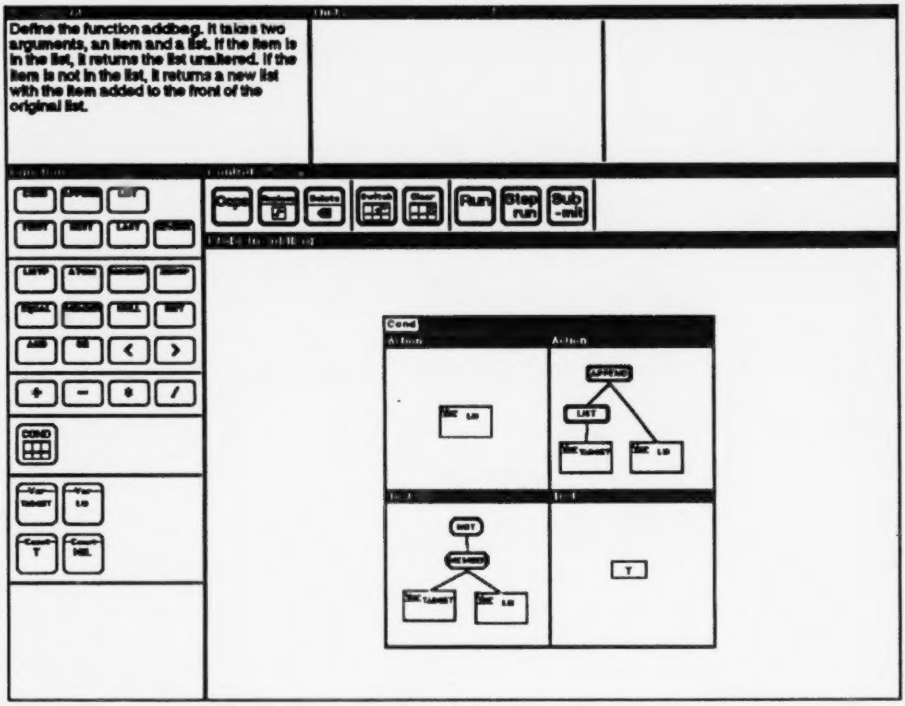
Another key feature in GIL is that students can see the data manipulated as they construct the program. In the early part of the curriculum, students indicate how a program operates on a particular example by specifying the data as it travels through the program as output from one function and input to another. In the more advanced curriculum described in this paper, students do not input the data themselves but can run a partially complete program on an example and inspect the data at any point in the program. Thus, students are led to test their understanding of the program's behavior as they construct it. Figure 1 shows a completed solution in GIL, and the corresponding text form of this program.

The GIL interface now includes 24 LISP programming constructs, including functions for extracting items from lists, constructing lists, arithmetic functions, predicates, logical functions, and conditionals. Students construct a graph by dragging a desired function into the graph, and drawing one or more links between the new function and others already in the graph. A link from a function to another function above it represents the flow of data—the output of one function is input to the function above it, e.g., the output of the LIST is input to the APPEND in the second Action of Figure 1.

The representation of the conditional structure is designed to make the structure of the program more explicit than equivalent text constructs. Conditional expressions are represented as a set of one or more pairs of Test-Action boxes, each corresponding to one clause of a conditional expression (see Figure 2). The student can drag a new Test-Action pair into the workspace and attach it to any pairs already present. LISP evaluates the clauses in a conditional in order (from left to right in a GIL graph), stopping on the first Test which succeeds. Reasoning in this domain is complicated by the need to have strategies for ordering conditional clauses so the program behaves correctly in all the target situations. GIL contains facilities for deleting the contents of a Test or Action, switching two Test-Action pairs or their components, and deleting a Test-Action pair.

In the conditionals curriculum, the student learns to use variables to represent inputs. A button is provided in the menu for each variable the problem will require. The student selects a name for the variable the first time he or she uses it. A conditional is shown in Figure 2. In this problem, called successor, the student has named the two variables TARGET and LIS to hold the target item and a list being searched, respectively. The problem also requires use of two constants, the atoms NO and END to be returned as the result in the first and second cases, respectively.

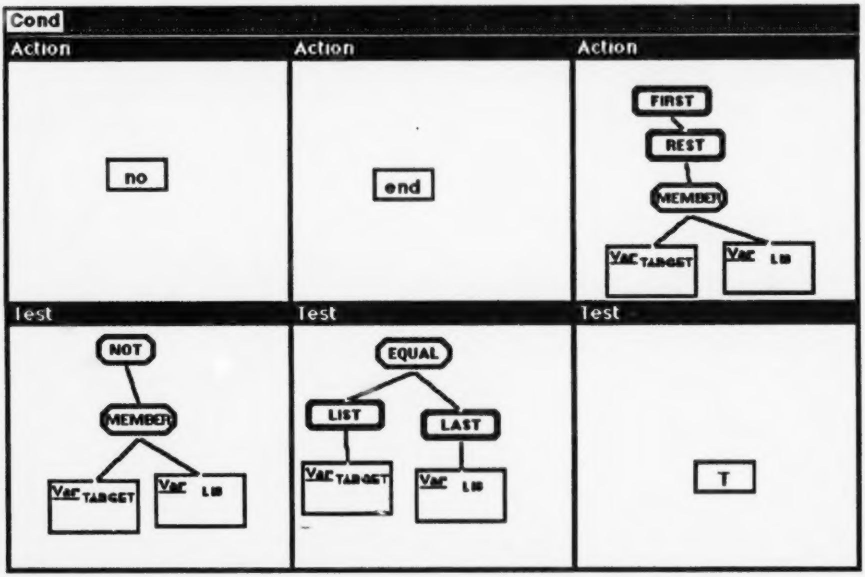
GIL encourages students to test their understanding of programs as they construct them. In the early curriculum, students make predictions about how the program will behave by specifying the data that are input and output at each point in this program. In the conditionals curriculum, students can *run* their programs at any point in their construction. To do this, the student assigns a particular data value to each variable and then observes the propagation of data through the program. GIL evaluates the components of the program as LISP does. Each node in the graph blinks as it is processed, and each Test and Action box greys out as GIL evaluates its contents. The intermediate results of each Test and Action are displayed on the title bars of the corresponding box, and the final result of the COND is displayed on its title bar. If GIL encounters an illegal expression that prevents running a portion of the program, it prints a diagnostic message in the Hints window. As data pass along each link in the graph, the link darkens. The student can then inspect the data that are passed along that link by clicking on it with the mouse. In this way, major intermediate products are automatically displayed, and more detailed results can be obtained on demand. Figure 3 displays the result of a run, in which the second test returned T, and thus the second action was selected (returning the constant END). The student also examined an intermediate data product within the second test (the output of LIST that is passed as input to EQUAL).



Define the function addbag. It takes two arguments, an item and a list. If the item is in the list, it returns the list unaltered. If the item is not in the list, it returns a new list with the item added to the front of the original list.

Run
Step run
Sub -mit
Cond
Action
Action
APPEND
LIST
Var TARGET
Var LIB
NOT
MEMBER
Var TARGET
Var LIB
T
COND
Var TARGET
Var LIB
Const T
Const NIL

Figure 1. A student's correct solution in GIL



Cond
Action
Action
Action
no
end
FIRST
REST
MEMBER
Var TARGET
Var LIB
Test
Test
Test
NOT
MEMBER
Var TARGET
Var LIB
EQUAL
LIST
LAST
Var TARGET
Var LIB
T

Figure 2. Example of GIL Conditional Structure

In summary, the important features of this extended interface are:

1. The flow of control and data is made explicit during the run of a program, and a trail is left of the path taken.
2. Local intermediate products are available on demand.
3. Constructing, evaluating, and debugging the program are integrated, since the student can run any portion of the partially complete program and input whatever data desired.

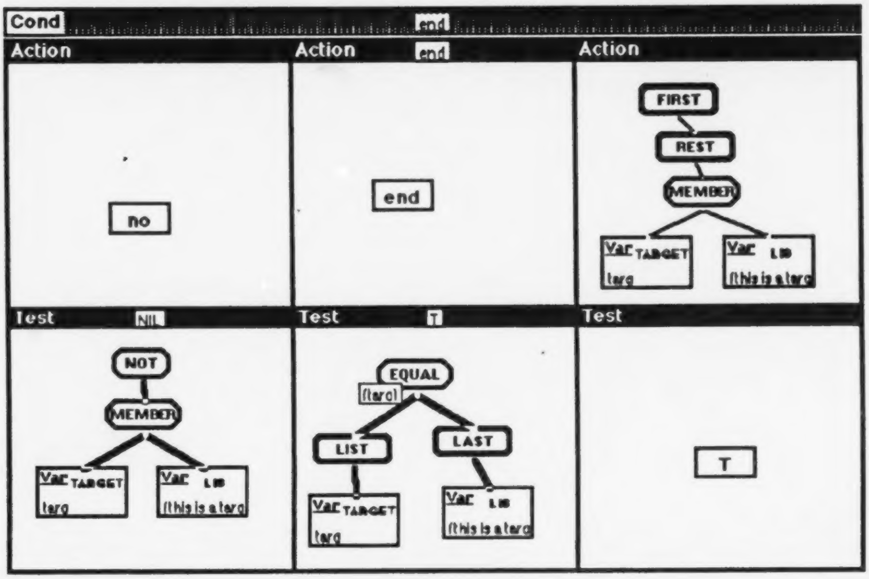

Cond
end
Action
Action
end
Action
no
end
FIRST
REST
MEMBER
Var TARGET
targ
Var LIS
(this is a targ
Test
NIL
Test
T
Test
NOT
MEMBER
Var TARGET
targ
Var LIS
(this is a targ
EQUAL
(targ)
LIST
LAST
Var TARGET
targ
Var LIS
(this is a targ
T

Figure 3. The result of running a program in GIL

Empirical Comparisons of GIL and Traditional LISP

We expect students will learn more effectively from GIL than from traditional LISP instruction. The process of program construction is designed to be a closer fit with the students” reasoning. The visual representation is designed to encourage students to test their understanding of a solution as they construct it. In addition, the simpler and more obvious syntactic requirements of the graphical representation help alleviate the potential processing overload in learning to produce syntactically correct programs while also learning the semantics and strategies for constructs in this domain. Finally, the graphical structure helps make the state of the system more available (particularly during runs of programs), lessening working memory load.

We have now begun a series of experiments that examine these proposed benefits of reasoning-congruent representations, as embodied in GIL. In this section, we will present some preliminary findings of a pilot comparison of students learning to program. This pilot study compares five students learning to program using GIL with eight students using a standard LISP interpreter. All students read three chapters of the same textbook (Anderson, Corbett, & Reiser, 1987) and solved the same problem sets in each chapter. The Text LISP students worked in a standard Common LISP environment with a simple screen editor provided to construct function definitions. The GIL students solved a subset of the same problems (some problems from Chapters 1 and 2 were excluded in GIL) but worked within GIL rather than a standard LISP interpreter. For the purposes of comparison, the version of GIL used contained only the graphical programming facilities described in this paper and did not provide any explanatory feedback. Thus, any benefits are due to the graphical representations and interface and not to intelligent feedback and guidance. We will discuss several differences that appeared in the two groups' solutions of the 11 problems in this chapter.

First, we examined the time required for the two groups to successfully solve the assigned problems. As expected, the GIL group solved the problems much more quickly than the control group–approximately 2 hours for GIL students versus 4.2 hours for the Text LISP group. Thus, there is a clear learning time benefit in the GIL representation.

More important comparisons, however, concern what the students understand about programming as a result of their learning sessions. First, we examined the types of changes made by students in each group to partial or complete solutions. In both the GIL and the Text LISP conditions, there are two types of changes possible: syntactic and algorithmic. A syntactic change in the Text LISP group entails moving parentheses or adding quotes, while an algorithmic change affects what the program does, such as replacing a programming construct or reordering the cases of a conditional. Syntax changes were rare among GIL students–the few instances were students forgetting that a function needs two arguments and only giving it one, then fixing the omission. Algorithmic changes in GIL students consisted of adding functions, reordering Test-Action boxes, etc. The Text LISP condition averaged 25 syntax changes, and 19 algorithmic changes per subject. The GIL group made 2 syntax changes total, and averaged 21 algorithmic changes per subject. Thus, while GIL clearly alleviates the overhead of syntactic issues in programming, the GIL students appear to require as many changes to their algorithms as do the Text LISP students.

One interesting measure of how well students understand their programs concerns "dead code." Dead code is a part of a program which will never be executed, or will never change the output of the program. An example is `(and (listp variable))`, in which the `and` does not change what its argument (the `listp` expression) returns. The Text LISP group had an average of 1.67 problems per subject in which dead code appeared. In contrast, the GIL group had an average of only 0.5 problems per subject with dead code. Moreover, there were no cases where a Text LISP student found and removed the dead code, while the GIL students who had dead code often removed it.

We also examined when students made changes in their programs. Students could change a program after running it (either in GIL or in the LISP environment), or they could make changes after simply looking at the current version of their solution. If the students make changes after simply examining their current solution, it suggests they have some prediction about how it will behave. Interestingly, only one student in the Text LISP group made changes after simply looking at a solution. The strategy of the Text LISP students seemed to be to type in a possible solution, test it, and then respond to any errors which the LISP environment pointed out. In contrast, all the GIL subjects, at one time or another, made changes after simply looking at their partial solutions.

Finally, we considered whether the groups differed in how they tested their programs. We examined how often students tested partial solutions. We expected the GIL students to have a better understanding of how their program will behave, and to be more focused on the goal of making predictions and tests of its behavior. Therefore, we expected GIL students to test their solutions as they construct them, while Text LISP students would construct a complete solution and then just attempt to test the entire solution rather than focusing on testing each component and understanding how the solution components fit together. As predicted, GIL students tested their partial solutions more frequently than the Text LISP solutions (4.3 versus 2 partial solutions per subject).

Summary and Conclusions

In summary, the results from this small pilot test suggest that the reasoning-congruent representations in GIL do facilitate students' understanding of programming. The higher occurrence of dead code in the Text LISP group, and its relative lack of removal, suggests that these students did not have as clear an understanding of the internal behavior of their programs as did the GIL students. In fact, the Text LISP students frequently copied a solution which had worked in a previous problem, and then modified it to work on the current problem. When a solution didn't work, these students often incrementally modified various parts of the solution in attempt to get it to run. The dead code often resulted from the portion of the previous solution which was unnecessary in the current solution. This strategy did allow the Text LISP students to worry less about syntax, and concentrate more on algorithmic concerns. However, it is clear from the change data that this strategy was not completely successful, in that half of the changes made by the Text LISP students were still syntactic. Also, these students usually made changes only when the environment told them to, and did not isolate parts of the program which might be wrong. On these fronts, GIL seems to provide clear advantages over the standard LISP instruction.

The present results do not provide strong evidence that the planning process itself for GIL students was easier (because both groups made approximately equal numbers of algorithmic changes). However, the results do suggest GIL students have a better understanding of the state of their program at any given point. Furthermore, disentangling the learning of programming syntax from the semantics and algorithm knowledge has clear benefits for learning time, and the lessened demands on working memory may help students adopt a more sensible strategy than iteratively transforming a solution until it is correct. In future investigations, we plan to minimize the syntactic aspects of programming for the Text LISP group as well, through a simple structured editor, to enable us to better isolate the source of the benefits of GIL's graphical representations.

Acknowledgements: We are grateful to Michael Ranney, Dan Kimberg, and Marsha Lovett for their contributions to the design of GIL, and to Shari Landes for assistance in data collection and analyses. This research was supported in part by contract MDA903-87-K-0652 from the Army Research Institute, by a research grant from the James S. McDonnell Foundation, and performed using equipment donated by the Xerox Corporation University Grant Program. The views and conclusion contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of these institutions.

References

Anderson, J. R., Boyle, C. F., & Reiser, B. J. (1985). Intelligent tutoring systems. *Science*, 228, 456-462.

Anderson, J. R., Corbett, A. T., & Reiser, B. J. (1987). *Essential LISP*. Reading, MA: Addison-Wesley.

Collins, A. & Brown, J. S. (1988). The computer as a tool for learning through reflection. In H. Mandl & A. Lesgold (Eds.), *Learning issues for intelligent tutoring systems*. New York: Springer-Verlag.

Larkin, J. & Simon, H. (1987). Why a diagram is (sometimes) worth ten thousand words. *Cognitive Science*, 11, 65-99.

Reiser, B. J., Ranney, M., Lovett, M. C., & Kimberg, D. Y. (1989). Facilitating students' reasoning with causal explanations and visual representations. In D. Bierman, J. Breuker, & J. Sandberg (Eds.), *Proceedings of the Fourth International Conference on Artificial Intelligence and Education* (pp. 228-235). Springfield, VA: IOS.

Reiser, B. J., Friedmann, P., Gevins, J., Kimberg, D. Y., Ranney, M., & Romero, A. (1988). A graphical programming language interface for an intelligent LISP tutor. *Proceedings of CHI-88 Conference on Human Factors in Computing Systems* (pp. 39-44). New York.

Reiser, B. J., Kimberg, D. Y., Lovett, M. C., & Ranney, M. (1991). Knowledge representation and explanation in GIL, an intelligent tutor for programming. In J. Larkin, & R. Chabay (Eds.), *Intelligent tutoring systems and computer assisted instruction: Shared issues and complementary approaches*. Hillsdale, NJ: Erlbaum.

Beyond the Centralized Mindset

Mitchel Resnick
MIT Media Laboratory
Cambridge, MA 02139 USA

Abstract: In many areas of science and social science, there is a growing interest in self-organizing systems—that is, systems in which orderly patterns arise without centralized control. But there is a problem: People generally have difficulty reasoning about such systems. In particular, people tend to assume centralized control even where none exists—a phenomenon I call the *centralized mindset*. To examine how people think about self-organizing systems (and to help them develop new ways of thinking about such systems), I developed a computational environment (called *Logo) that people can use to create and explore models of self-organization. Using *Logo, people can write simple programs for thousands of interacting "creatures," then observe the group behaviors that emerge. This paper describes the design and goals of *Logo, and it discusses initial empirical studies of high-school students using *Logo.

Introduction

In many areas of science and social science, there is a growing interest in *self-organizing systems*—that is, systems in which orderly patterns arise without any centralized control. An ant colony, for example, uses sophisticated strategies to search for food, but no single ant (or group of ants) develops or manages the strategies. Rather, foraging patterns emerge through the interactions of thousands of ants, none of which has a sense of the overall design (see, for example, Deneubourg et al., 1986).

Examples of self-organization are seemingly everywhere: evolution, market economies, and traffic jams can all be explained in terms of self-organizing behaviors. Nevertheless, people seem to have great difficulty reasoning about self-organizing systems. In particular, people tend to assume centralized control where none exists.

In some ways, this "centralized mindset" is not surprising. Many phenomena in the world are, in fact, organized by a central designer. When people see neat rows of corn in a field, they assume (correctly) that the corn was planted by a farmer. When people watch a ballet, they assume (correctly) that the movements of the dancers were planned by a choreographer. And from an early age, many children participate in systems (such as school classrooms) where power and authority are centralized (often excessively so).

On the other hand, the centralized mindset can often cause misconceptions. When people see a flock of birds, many of them assume (incorrectly) that there must be a leader of the group. And when people see the diversity of complex creatures in the world, many of them assume (incorrectly) that there must be a single Designer of Life.

This paper describes a preliminary investigation into self-organizing systems and the centralized mindset. As part of this research, I have developed a new computer-based environment called *Logo (pronounced *star-logo*). Using *Logo, people can write simple programs for thousands of interacting "creatures," then observe the collective behaviors that emerge. *Logo is designed to help people develop new ways to think about self-organizing systems—going beyond the centralized mindset. In this paper, I describe the design and goals of *Logo, and I discuss initial work with high-school students using *Logo.

Background

My research on *Logo and self-organization has been motivated, in part, by my previous work on LEGO/Logo (Resnick Ocko, and Papert, 1988). LEGO/Logo is a computer-controlled construction set, in which students write Logo programs to control machines that they build out LEGO pieces (including gears, motors, and sensors). Elementary-school students have used LEGO/Logo to build and program a wide assortment of creative machines, including a "chocolate-carob factory" (inspired by the Willy Wonka children's stories) and a machine that sorts LEGO bricks according to their lengths.

During the past several years, members of our research group (Resnick and Martin, 1990; Granott Farber, 1990) have paid special attention to ways in which students think about LEGO "creatures" (that is, LEGO/Logo machines designed to look like and act like living creatures). For example, we have investigated how students describe their creatures—sometimes using psychological descriptions (*It wants to go towards the light*), and and other times using mechanistic descriptions (*When the light hits this sensor, it sends a signal to the motor*). We have observed students acting somewhat like researchers in the new field of Artificial Life (Langton, 1989), attempting to understand living systems by building models of them. Indeed, we have found that LEGO creatures provide a good context for students to think about and learn about some of the central ideas of Artificial Life, such as *feedback* and *levels of organization*.

Unfortunately, certain core ideas of Artificial Life research (such as *self-organization* and *emergence*) are not easily explored with our current LEGO technology. For example, it would be very difficult to simulate an ant colony with LEGO robots. So I decided to go back to the computer screen, creating a simulated ecosystem (called *Logo) with thousands of graphic creatures.

Like LEGO/Logo (and Logo itself), *Logo is based on the idea that people learn with particular effectiveness when they are engaged in creating personally meaningful artifacts. Papert (1986) has termed this idea *constructionism*. In using *Logo, people don't just observe or manipulate self-organizing systems; they design the systems themselves. This constructionist approach might be particularly important in learning about self-organizing systems. People observe self-organizing systems in the world all of the time, but the experience is largely *passive*. Rarely if ever do they *actively design* self-organizing systems. In providing people with the opportunity to do so, *Logo aims to help people develop new models for thinking about patterns and control.

Methodologically, this research is different from many other educational-research projects. In many educational studies, researchers begin with a clear idea of the *right* way to think about a particular subject, then they try to help students learn how think in that way. Unfortunately, there are no strong theories on how to think about self-organizing systems. Thus, my work with *Logo has the dual goals of developing new intellectual tools, while also studying how people think about and appropriate those tools.

To my mind, these goals are inseparable. In developing new intellectual tools (and new computational tools), it is critically important to understand how people might make use of (and make sense of) those new tools. Developing new ways of thinking requires an understanding of both the *subject* (the content) and the *subjects* (the people who do the thinking).

Design of *Logo

There is an old saying that goes something like this: If a person has only a hammer, the whole world looks like a nail. Indeed, a person's perceptions and models of the world are strongly shaped by the objects that exist in the world. The same is true for computational systems. The way people interact with (and think about) a computational system depends strongly on the *objects* that compose the system. If the objects are well-chosen for the intended task, even novices will use the system productively. If the objects are not well-chosen, even experts will struggle.

*Logo is designed around two primary types of objects, chosen specifically to facilitate the modelling of self-organizing systems and emergent phenomena. One type of object is a *creature*. Traditional versions of the Logo programming language include a graphic "turtle" on the screen. Students can command the turtle to move and draw lines on the screen (Papert, 1980). *Logo creatures go beyond the standard Logo turtle in several ways. Most important, *Logo programmers can control *thousands* of creatures, and all of the creatures can act in parallel. In addition, *Logo provides programmers with built-in primitives to control local interactions between the creatures, and to create new creatures dynamically. Programmers can use *Logo creatures to represent birds or ants or molecules or automobiles—or any other type of object that moves in the world. By creating different "breeds" of creatures, programmers can simulate situations with multiple object types—for example, foxes and rabbits in an ecology model, or antibodies and antigens in an immunology model.

The second type of *Logo object, called a *patch*, is a more striking departure from traditional Logo (and other programming languages). A patch is a small section of the "environment" in which the creatures live. (The term *patch* is borrowed from Hogeweg (1989).) If, for example, the creatures are programmed to release a "chemical" as they move, each patch can keep track of the amount of chemical that has been released within its borders. Like creatures, patches are "active" objects, capable of computation. So, as the creatures move, each patch could diffuse some of its chemical into neighboring patches, or it could grow "food" based on the level of chemical within its borders. *Logo includes several special-purpose primitives (such as **follow-gradient**) to facilitate interactions between creatures and patches, and others (such as **diffuse**) to facilitate interactions between neighboring patches.

Thus, the environment is given equal computational status to the animals that inhabit it. Animals (in the form of creatures) and the environment (in the form of patches) are both computationally active objects, and they can mutually influence one another. Other "creature-oriented" programming languages (e.g., Steels, 1989) tend to treat the environment as a passive entity, manipulated by the creatures that move within it. This view, not surprisingly, matches the way many people view the Earth itself. By reifying the environment, *Logo aims to change the way people think about creature-environment interactions—perhaps leading to new and richer ways of thinking about how phenomena emerge in the world.

The current version of *Logo is implemented on the Connection Machine, a massively-parallel computer with thousands of processors (Hillis, 1985). In this implementation, each

Connection Machine processor controls one creature and several patches. But massively-parallel hardware is not theoretically important to the *Logo approach. *Logo aims to provide the programmer with a clear *conceptual model* for massive parallelism. The programmer writes programs *as if* the creatures and patches are all acting in parallel, regardless of the underlying implementation. (Indeed, we are currently porting *Logo to a sequential (and less expensive) computer, so that more people will be able to use it.)

The thinking underlying the design of *Logo is, in some ways, analogous to the thinking underlying the early versions of Logo itself. In the late 1960s, Logo aimed to make then-new ideas from computer science (like procedural abstraction and recursion) accessible to a larger number of users. Similarly, *Logo aims to make 1990s ideas from computer science (like massive parallelism) accessible to a larger audience. And whereas Logo included special features (most notably, the turtle) to facilitate explorations of mathematical ideas such as differential geometry (Abelson and diSessa, 1980), *Logo includes special features (creatures and patches) to facilitate explorations of self-organizing systems.

Empirical Studies

During the past year, a small number of high-school students have used *Logo, working in groups of two. Their activities fall into several categories: talk-through hypotheticals, what-if simulations, and user-designed projects. Examples (and analyses) of these activities are presented below.

Talk-Through Hypotheticals

At times, I presented students with hypothetical situations involving self-organizing phenomena, and I asked them to analyze the situations or design solutions. For example, I asked students to consider the following hypothetical situation (inspired by Steels (1990)):

> *Suppose that we discovered large deposits of gold on some distant planet. It is too dangerous and costly to send human astronauts to this planet, so we decide to send a spaceship with several thousand small robots. Each robot has a sensor to detect when it gets near gold, and a scoop to dig for (and carry) the gold. Once the spaceship lands on the planet, we want the robots to explore for gold and bring the gold back to the spaceship. How should we program each of the robots? In other words, what type of rules and strategies should the robots follow?*

I chose this problem since it is very similar to a problem in the natural world: How do ants find and collect food? Ants generally solve the problem in a very decentralized way. They begin by searching randomly. When an ant finds food, it carries the food back to the nest, emitting a chemical pheromone as it walks. If another ant happens to detect this pheromone trail, it follows the trail to the food source, then reinforces the trail on its way back to the nest.

In presenting the robots-searching-for-gold problem, I asked the students to suggest general rules and strategies for the robots. (I did *not* ask them to write actual computer programs for the robots.) The students developed a wide variety of strategies. But there were certain consistencies in the student responses. Certain ideas came up again and again:

- *Centralized decision-making.* In most of the strategies, one robot (or a small group of robots) was put in charge of the exploration process. For example: The "leader" tells the "explorer robots" where to explore. If an explorer finds gold, it reports back to the leader at the spaceship. The leader then assigns other robots to help the successful explorer.

- *Global communications.* The strategies generally assumed that each robot could communicate with every other robot, no matter where the robots were located, no matter how many robots were trying to communicate at the same time.

- *Absolute coordinates.* Many of the strategies assumed that the robots always had perfect knowledge of their locations. In one case, the "leader" assigned each robot a specific Cartesian coordinate to explore. Once a robot found gold, it communicated its exact location to the other robots (so that they could come and help).

These strategies conflict with actual ant strategies in almost every way. Ant strategies are decentralized rather than centralized; local rather than global; relative rather than absolute. That is not to say that the student strategies were *wrong*; in fact, many of the student strategies seemed like they would work, and some were very creative. But it is intriguing that the students were so committed to centralized approaches, even in a situation inspired by the decentralized strategies of ants. Some of the students had already seen *Logo simulations of ants foraging for food, but even they chose centralized strategies when presented with the robot design problem.

What-If Simulations

In some cases, I showed existing *Logo simulations to the students, and I asked them to predict what would happen if certain features or parameters were changed. One of the simulations was inspired by certain small insects that aggregate into clusters (Prigogine and Stengers, 1984). The mechanism for this behavior involves a chemical pheromone that the creatures emit. With each step, the creatures drop a bit of pheromone, but they also "sniff ahead" for the pheromone. If they can sense some pheromone, they follow the gradient of the pheromone (that is, they move in the direction of greatest concentration). The behavior of each creature can be simulated with a simple *Logo program:

```
to creature-rule
if ask patch-here [pheromone] > smell-threshold
        [follow-gradient pheromone]
demand patch-here [make "pheromone pheromone + 1]
right random 40 left random 40
forward 1
end
```

Meanwhile, each patch follows this rule: Let a little of your pheromone evaporate, then let the pheromone diffuse into neighboring patches.

```
to patch-rule
make "pheromone pheromone * .95
diffuse pheromone
end
```

If you start the simulation with a small number of creatures, not much happens. The creatures wander around randomly, trying to detect the pheromone. But the pheromone diffuses and evaporates too quickly. The creatures look somewhat like the particles in an ideal gas. But if you add enough creatures to the simulation, the behavior changes drastically. With lots of creatures, there's a better chance that a few creatures will wander near one another. When that happens, the creatures will collectively drop a fair amount of pheromone, creating a sort of pheromone "puddle." The pheromone in the puddle will be above the "smell-threshold," so the creatures in the puddle will follow the pheromone gradient. Thus, they stay within the puddle and drop even more pheromone—a type of positive feedback. If other creatures wander into the puddle, they are likely to join the group. Before long, there are clusters of creatures all over the screen.

This simulation suggests many "what-if" questions. For example: What would happen if the creatures could detect smaller amounts of pheromone? We can find out by reducing the value of **smell-threshold** in the **creature-rule** procedure, then running the simulation again. What happens? The creatures form more and smaller clusters, and they form the clusters more quickly. Interestingly, this result ran counter to the students' intuitions. Most of the students predicted (correctly) that clusters would form more quickly, but almost all students predicted (incorrectly) that the clusters would be larger than before.

Why did the students have this false intuition? Some of the students explained their thinking like this: the creatures are trying to form clusters, so if the creatures have a better sense of smell, they will do a "better job" and form larger clusters. What's the flaw? Creatures are not really trying to form clusters; they are simply trying to follow a pheromone gradient. The clusters are an emergent phenomenon, on a different "level" from the creatures. The students incorrectly assumed that someone or something must be "trying" to form the clusters.

It should be noted that such misconceptions are not limited to students or novices. At one point, while I was working on the aggregation simulation, Marvin Minsky wandered past and asked what I was doing. I explained that I was experimenting with self-organizing behaviors. Minsky looked at the clusters of creatures on the screen and shook his head: "That's not self-organizing. The creatures are just going after the food." He assumed that there must be some centralized source (food) attracting the creatures. Clearly, Minsky has done a great deal of thinking about decentralized systems (Minsky, 1988). And once I explained the program, he quickly understood what was happening. Nevertheless, presented with a novel situation, he resisted the suggestion that the situation might involve self-organization, and he opted for a centralized explanation instead.

User-Designed Projects

The students spent most of their time designing their own *Logo simulations. Many of their simulations were inspired, in part, by magazine articles about self-organizing phenomena which I had given to the students.

Two students (one of whom had recently received his driver's license) decided to design a simulation of traffic flow. They started with the simplest of rules. If a car sensed another car ahead of it, it slowed down. If it didn't sense another car, it sped up. (There was no speed limit.) With this simple rule, the students didn't expect much to happen. But when they started the simulation, the cars bunched into a realistic-looking traffic jam.

The students were surprised. Why should a jam form at all? The ensuing discussion highlighted the importance of small fluctuations. If all cars were evenly spaced, travelling with the same velocity, there would be no traffic jam. But small fluctuations (such as two cars spaced close together, or one car with a higher velocity) can ripple through the cars like a shock wave, causing a traffic jam. After some discussion, one of the students recognized that the formation of the traffic jam was analogous to the formation of clusters in the earlier example.

When the students started watching the traffic jam, they were in for another surprise: the traffic jam moved backwards. They found this behavior counter-intuitive. How can a traffic jam move backwards when all of the cars within it are moving forward? This behavior highlighted an important idea: emergent structures (like traffic jams and waves) often behave very differently than the elements that compose them.

Thinking About Self-Organization

This preliminary work with high-school students provides strong evidence for the prevalence of the centralized mindset. The students, in many of their comments and actions, assumed centralized control where none existed.

The work with students also provides initial hints to intellectual tools that are useful for thinking about self-organization in new ways. As I worked with students on *Logo simulations, a few *core concepts* arose again and again. Among them: feedback, critical density, randomness, diffusion, gradients, levels, local maxima and minima. Also, the students and I began to develop some qualitative *heuristics* that seemed helpful in thinking about self-organizing systems. A few examples:

- *Complex patterns can emerge from simple rules.*
- *Large-scale patterns can emerge from local rules.*
- *Randomness can promote stability and order.*
- *Systems don't necessarily act like their parts.*
- *Small fluctuations can lead to big changes.*

These heuristics are not very helpful in making definitive predictions about what will happen in a particular system. Indeed, they are generally of the form: "Don't be surprised by non-intuitive behavior." But these heuristics can provide an initial framework for "making sense" of self-organizing systems. Moreover, they can help in designing systems, making designers aware of an expanded number of ways to generate orderly patterns.

Clearly, this research effort is still in its early stages. One future direction is to develop a finer-grained understanding of the origin and evolution of the centralized mindset. Another research direction is to develop better heuristics, both for analyzing existing self-organizing systems, and for designing self-organizing behaviors for specific tasks.

Acknowledgments: Hal Abelson, Seymour Papert, Brian Silverman, and Ryan Evans have provided encouragement, inspiration, and ideas for the *Logo project. The LEGO Group and the National Science Foundation (Grants 851031-0195, MDR-8751190, and TPE-8850449) have provided financial support.

References

Abelson, H. and diSessa, A. (1980). *Turtle Geometry: The Computer as a Medium for Exploring Mathematics.* Cambridge, MA: MIT Press.

Deneubourg, J.L., Aron, S., Goss, S., Pasteels, J.M., and Duerinck, G. (1986). Random Behavior, Amplification Processes, and Number of Participants: How They Contribute to the Foraging Properties of Ants. *Physica D* 22: 176-186.

Granott Farber, N. (1990). Puzzled Minds and Weird Creatures: Spontaneous Inquiry and Phases in Knowledge Construction. In Harel, I. (ed.) *Constructionist Learning*. Cambridge, MA: MIT Media Laboratory.

Hillis, W.D. (1985). *The Connection Machine*. Cambridge, MA: MIT Press.

Hogeweg, P. (1989). MIRROR beyond MIRROR, Puddles of LIFE. In Langton, C. (ed.) *Artificial Life*. Redwood City, CA: Addison-Wesley.

Langton, C., ed. (1989). *Artificial Life*. Redwood City, CA: Addison-Wesley.

Minsky, M. (1988). *The Society of Mind*. New York: Simon & Schuster.

Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

Papert, S. (1986). *Constructionism: A New Opportunity for Elementary Science Education*. Proposal to the National Science Foundation. Cambridge, MA: MIT Media Laboratory.

Prigogine, I. and Stengers, I. (1984). *Order Out of Chaos*. New York: Bantam Books.

Resnick, M., Ocko, S., and Papert, S. (1988). LEGO, Logo, and Design. *Children's Environments Quarterly* 5 (4): 14-18.

Resnick, M. and Martin, F. (1990). *Children and Artificial Life*. Epistemology and Learning Memo 10. Cambridge, MA: MIT Media Laboratory.

Steels, L. (1990). Cooperation between Distributed Agents through Self-Organization. In Demazeau, Y. and Muller, J.P. (eds.) *Decentralized A.I.* Amsterdam: North-Holland.

Steels, L. (1989). *The RDL Manual*. AI Memo 89-11. University of Brussels.

An Architecture for the Social Construction of Knowledge

Marlene Scardamalia
Centre for Applied Cognitive Science
Department of Measurement, Evaluation and Computer Application
252 Bloor St. West, Toronto, Ontario M5S 1V6

Abstract: CSILE (Computer-Supported Intentional Learning Environments) is a hypermedia system built around a student-generated database. In CSILE, the basic knowledge object is a note/idea. Notes are the objects of inquiry and are compiled into note complexes within knowledge-building environments. Students work cooperatively to elaborate and upgrade information in line with the support systems within different knowledge-building environments–including data exploration, explanatory coherence, "how-it-works," analogy, and publication environments. The idea of a knowledge-building environment is that knowledge is brought into the environment and something is done to it that enhances its value. Tools available in the environment are means by which the value-adding work is done. Consequently, the design of environments and tools is guided by the goal of maximizing the value added to knowledge–either the public knowledge represented in the community database or the private knowledge and skill of the user.

The concept of a student-generated database at the heart of CSILE is made more dominant by distinguishing online but external databases from the student-generated database. The protocol for importing information from a wide range of electronic media into the student database involves cooperation and specialized processing of information. We pursue the knowledge-value-added principle here, also, so that material is not entered without the information having been processed and enhanced in some way by the student.

To support the social construction of knowledge a knowledge-building environment must provide a reason to use the tools, a context in which their use contributes to the overall development of the community's knowledge, and supportive means both for translation of information from notes into input to the tools and translation of output from the tools into notes. Most importantly, the instructional program as a whole must be designed in such a way that systematic pursuit of problems of meaning, explanation, "how-it-works," etc., are integral to the curriculum.

Promoting Abstract Strategies in Algebra Word Problem Solving

Mark K. Singley
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

John R. Anderson
Carnegie-Mellon University
Pittsburgh, PA 15213

Jody S. Gevins
NASA Ames Research Center
Moffett Field, CA 94035

Abstract: We describe an experiment comparing three abstract strategies for algebra word problem solving in terms of both learning and transfer. The skeletal strategy is a working-forwards, expert strategy that most closely approximates standard textbook instruction. The means-ends strategy is a variant of means-ends analysis that begins with the goal and works backwards. The diagram strategy makes heavy use of a diagram to control the generation and combination of constraints. We found that the diagram strategy was superior to the skeletal strategy in terms of learning, and that the means-ends strategy was superior in terms of transfer. Our approach stresses the importance of explicit task analysis as a prerequisite to promoting transfer.

The New Realism in Transfer Research

In the last decade, some exciting progress has been made in codifying and teaching moderately general skills. This represents a turnaround of sorts of the historical trend which shows a depressing lack of evidence for the notion of general transfer (for a review, see Singley and Anderson, 1989). First, in the domain of mathematics, Schoenfeld (1985), after numerous failed attempts by other researchers, has been moderately successful in teaching abstract heuristic strategies to college students. His basic strategy has been to take the heuristics of Polya as a starting point and to elaborate them with more specific rules and guidelines. In physics, Larkin (1989) and her colleagues have codified general principles and mechanisms of interpretationwhich apply across subdomains like fluid statics and electricity. In computer programming, Klahr and Carver (1988) taught an abstract debugging strategy to elementary school children and observed transfer to the debugging of plans outside of a programming context. Finally, Brown and her colleagues (Palincsar and Brown, 1984) have successfully taught general comprehension strategies that apply broadly across contexts. What characterizes these successful studies, and how do they differ from previous unsuccessful attempts? These studies share three important and inter-related features:

Importance of explicit task analysis. The above studies all stress the importance of a thorough and accurate task analysis as a necessary precondition to teaching for transfer. In the cases of Larkin (1989) and Klahr and Carver (1988), these task analyses were expressed as simulation models which could perform the task, which is the ultimate check on the sufficiency of a proposed method. Although simulation models are neither a necessary nor sufficient condition for success, the kind of explicit task analysis that they require does seem to drive out the unsubstantiated wishful thinking that characterizes much of the previous work on transfer.

Appreciation of computational demands. One consequence of a serious attempt at task analysis is an enhanced appreciation of how difficult it is to apply abstract knowledge across contexts. This leads to moderated expectations about what kinds of transfer are possible, and also to an enhanced appreciation of the importance of strategic and/or control knowledge. In defining an abstract strategy, the researcher must be explicit about when the strategy is useful, and when it is not. An inability to specify these conditions is a sure sign that the strategy is deficient.

Onus on the teacher, not the student. The message from the above studies is that, not only must the teacher scope out the domain and codify the abstract strategy, the teacher must also successfully communicate that strategy to students. Both of these steps are difficult and prone to error. It seems, however, that students should not be expected to induce these abstractions entirely for themselves if transfer is the desired outcome.

The Present Work

In this paper we report on an attempt to teach abstract strategies in the domain of algebra word problem solving. Like the above studies, we stress the importance of an explicit task analysis, and teach the products of that analysis directly to students. In fact, these two concerns are forged together in our methodology. We use a model-tracing intelligent tutoring system for algebra word problems (Singley, Anderson, Gevins, and Hoffman, 1989) to teach our students. This means that (1) our strategies are specified in the form of running simulations, and (2) these strategies are enforced (and reinforced) by the design of the system interface (a proactive influence) and by system responses to student actions (a reactive influence). We will now describe our codification of strategies for algebra word problem solving, the system we used to teach those strategies, and an experiment we conducted to assess the effectiveness of those strategies.

Task Analysis of Algebra Word Problem Solving

We will develop our task analysis in the context of an example problem, the classic picture frame problem:

A picture frame measures 20 cm by 14 cm. 160 sq cm of picture shows. What is the width of the frame?

Generally, to solve an algebra word problem, one must bring to bear a set of contraints which serve to express the answer of the problem in terms of known quantities. Our task analysis claims that word problem solving is difficult because it is a knowledge intensive task that involves a number of difficult components, not the least of which is the search through the space of constraints (we view a constraint as a primitive equation that relates two or more problem elements). Indeed, data from an initial study we conducted showed that students make ten times as many errors on word problems that involve multiple constraints, and that the difficulty of a problem is directly related to the number of primitive equations that make up the solution. This is consistent with the results of the National Assessment of Educational Progress, a national survey of mathematical problem solving abilities (Carpenter, Corbitt, Kepner, Lindquist, & Reys, 1980). The survey showed that performance on one-step word problems was quite good, but that performance on multi-step problems was quite bad.

These results suggest that students need help managing their search processes, and this is what we chose to focus on. Through our task analysis, we defined three distinct strategies for the generation and combination of constraints:

Skeletal strategy. The skeletal strategy is a working forwards strategy that most closely approximates the kinds of example solutions presented in textbooks (Forester, 1984). In the skeletal strategy, one first writes the primitive equation that is in some sense key to solving the problem. This primitive equation in most cases does not contain the goal variable, and its presentation is entirely unmotivated except for the fact that it *works*. For example, to solve our example problem, one would first recognize that the area of the picture is equal to the length of the picture times the width of the picture. One would then realize that the length of the picture equals $20 - 2x$, and the width of the picture equals $14 - 2x$, and suddenly one has an equation that is solveable for x:

$$(20 - 2x)(14 - 2x) = 160$$

The skeletal strategy gets its name from the fact that the first equation written is the easily-identified skeleton of the final equation. It is easily identified because the skeletal equation (which again typically does not contain the goal variable) produces a parse tree for the final equation that has minimal depth and maximal breadth. Thus, the skeletal strategy generates final equations whose structures are easily understood. This may be why it is popular in textbooks: it provides the most easily understood post-mortem of how a problem was solved. However, the strategy is not generative in that particular skeletal equations and subsequent solution steps are largely unmotivated. The search problem is totally ignored; learning is reduced to the more-or-less rote acquisition of specific problem-solution patterns or schemas.

Means-ends strategy. Means-ends analysis is a working backwards strategy that has been well-documented as a novice strategy for solving systems of equations. However, to our knowledge, it has never before been taught directly to algebra students. In means-ends analysis, one first writes a primitive equation that contains the goal variable. Given this equation, subgoals are set to find values for the remaining unknowns. Additional constraints are generated which contain the unknowns and are substituted into the equation containing the goal. The problem is solved when the goal variable is stated in terms of knowns. The advantage of means-ends analysis over the skeletal strategy is that the first as well as all subsequent steps are well-motivated. However, the resulting final equation is difficult to understand because the strategy generates a tree structure that has maximal depth and minimal breadth. Thus, it is hard to understand the solution once it is generated. Indeed, Sweller, Mawer, & Ward (1983) have claimed that means-ends analysis interferes with learning because it makes heavy demands on working memory and obscures the underlying problem schemas. Nevertheless, when the goal of instruction is not the acquisition of a catalogue of schemas but rather an abstract strategy that applies broadly across problem types, means-ends analysis seems superior to the skeletal strategy. See Figure 1 for a comparison of the means-ends and skeletal strategies on the example problem.

Diagram strategy. Another strategy we have defined makes direct use of the diagram. It involves the repeated labelling of objects in the diagram using only the goal variable and known quantities until an equation can be written that makes use of the labels and contains the goal variable. For example, Figure 2 shows the labelling of a diagram for the picture frame problem using this strategy. At this point, an equation can be written that states the area of the picture (160) in terms of the length of the picture ($20 - 2x$) and the width of the picture ($14 - 2x$). The virtue of this strategy is that it introduces no extra variables and thus places a smaller burden on working memory. In addition, the diagram provides heuristic guidance about which object to label next, so the search problem is somewhat mitigated.

Means-ends

$20 = l + 2x$
$l = 160/w$
$20 = 160/w + 2x$
$w = 14 - 2x$
$20 = 160/(14 - 2x) + 2x$

Skeletal

$l * w = 160$
$l = 20 - 2x$
$(20 - 2x)w = 160$
$w = 14 - 2x$
$(20 - 2x)(14 - 2x) = 160$

Figure 1. Comparison of means-ends and skeletal strategies.

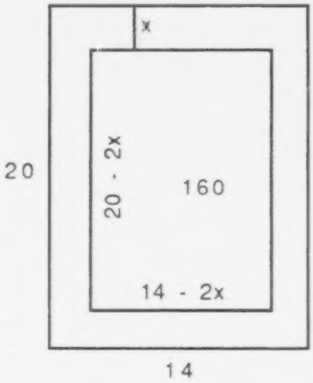


Figure 2. The diagram strategy.

The drawback of the diagram strategy is that the termination point is unclear. One must continually monitor the diagram to see when the final equation can be written. Also, the diagrams themselves vary between problem types, and the strategy is not enabled until a correct diagram is constructed. This limits the generality of the strategy somewhat.

Experiment: Comparison of Strategies

We set out to evaluate the three strategies in terms of both learning and transfer. The experiment involved six training conditions: three strategies by two levels of enforcement. Strict enforcement meant that students were not allowed to deviate from the prescribed strategy while solving problems with the tutor. Loose enforcement meant that students could perform any legal step in the problem space, regardless of whether it fit in to a particular strategy.

Method

Subjects

Subjects were 24 seventh-grade children who were concurrently taking their first course in algebra. All had had some exposure to word problems, and had been taught how to solve linear and quadratic equations.

Materials

Subjects were assigned to one of four training problem schemas: coin mixture, liquid mixture, rectangular frame, or two-sided frame. All training problems were drawn from a single schema. Following training, subjects solved problems from each of the other three types, in addition to more problems of the same type. The other three types of problems represented near and far transfer problems. For example, if a subject were trained on coin mixture problems, then the liquid mixture problems would represent near transfer problems, and the two types of frame problems would represent far transfer problems. Near transfer is defined by the sharing of some (but not all) of the underlying primitive equations that define a problem schema. Far transfer is defined by disjoint sets of underlying primitive equations.

Procedure

Subjects first took a short pretest involving algebraic manipulation, e.g., solving linear equations and simplifying expressions. They then read short (about 10 pages) booklets describing one of the three strategies. Each booklet demonstrated the use of the strategy on a single problem. The experimenter then demonstrated the use of the tutor on two sample problems. Both demonstrations involved application of the strategy. Then subjects solved problems drawn from one of the four schemas using the tutor until a training criterion was achieved: two consecutive problems with a total of two or fewer errors and no help requests. Then subjects saw a sequence of transfer problems. (In the transfer phase of the experiment, the strict enforcement of strategies was lifted.) First, subjects solved two more problems drawn from the training schema, but involving different configurations of givens and goals than they saw during training (within-schema transfer). Next, subjects solved two near-transfer problems. Finally, subjects solved four far-transfer problems, two of each type.

Results

Figure 3 shows the mean number of trials to criterion for each of the six conditions. An analysis of covariance using pretest score as a covariate showed a marginal main effect for type of strategy ($F = 2.7$, $df = 2$, $p < .1$). A post-hoc t-test revealed that the diagram strategy took significantly fewer trials to reach criterion than the skeletal strategy ($t = 2.32$, $df = 16$, $p < .05$). Besides the marginal main effect, the analysis of covariance yielded a marginal interaction between strategy and level of enforcement ($F = 2.3$, $df = 2$, $p < .15$). At the risk of overinterpretation, this result suggests that, whereas generative strategies like the means-ends and diagram strategies benefit from loose enforcement, the more routinized skeletal strategy does not. Other analyses of covariance revealed that loose enforcement results in fewer errors per problem during training (2.8 vs. 4.1; $F = 4.5$, $df = 1$, $p < .05$) and marginally more help requests (2.3 vs. .8; $F = 3.9$, $df = 1$, $p < .07$). Comparisons of the strategies on these dependent measures yielded no significant effects.

Figure 4 shows comparisons of the strategies on the transfer problems in terms of help requests per problem. Other dependant measures, such as time per problem and errors per problem, yielded no significant effects. We have collapsed across levels of enforcement, as this training factor also had no effect on transfer performance. What we see in terms of help requests per problem, however, is the emergence of the means-ends strategy as the superior strategy on both near and far transfer problems. Analyses of covariance and subsequent t-tests revealed that subjects taught the means-ends strategy asked for help significantly fewer times than subjects taught the skeletal strategy in both the near transfer ($t = 3.2$, $df = 16$, $p < .05$) and far transfer ($t = 2.7$, $df = 16$, $p < .05$) problems.

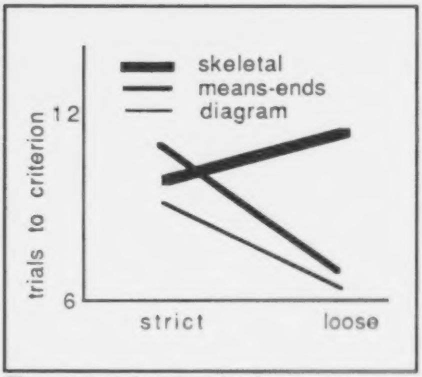


Figure 3. Trials to criterion.

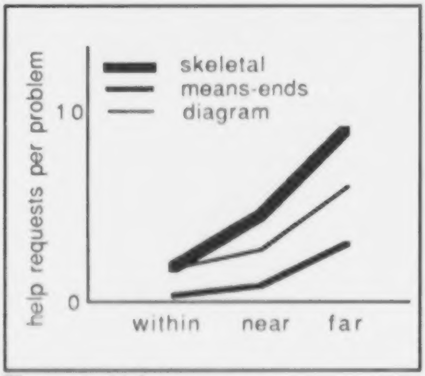


Figure 4. Help requests.

General Discussion

We have demonstrated the superiority of the diagram and means-ends strategies over a formalization of the standard textbook strategy in terms of both learning and transfer. The diagram strategy leads to the fastest criterion performance, and the means-ends strategy leads to the fewest requests for help in both near and far transfer situations. It is interesting that the same strategy was not superior in terms of both learning and transfer. The Gestalter Katona (1940) first hinted at this tradeoff between learning and transfer when he observed that a rote strategy for solving matchstick problems was easier to learn but showed less transfer than a more meaningful strategy. Singley and Anderson (1989) showed that, in production system simulations, general methods require more rules than specific methods, which again implies the learning-transfer tradeoff. These results place an important restriction on the work of Sweller and his colleagues (Sweller, Mawer, and Ward, 1983). Sweller's position is that means-ends analysis is a poor strategy for novices, in that it places undue demands on working memory and interferes with schema acquisition. Given the present results, however, perhaps we should question whether schema acquisition is a proper goal of instruction.

References

Carpenter, T.P., Corbitt, M.K., Kepner, K.S., Lindquist, M.M. & Reys, R. (1980). Results of the second NAEP mathematics assessment. *Mathematics Teacher*, 329-338.

Forester, P.A. (1984). *Algebra I*. Menlo Park, CA: Addison Wesley.

Katona, G. (1940). *Organizing and memorizing*. New York: Columbia University Press.

Klahr, D. & Carver, S. (1988). Cognitive objectives in a LOGO debugging curriculum: instruction, learning, and transfer. *Cognitive Psychology*, 362-404.

Larkin, J. (1989). What kind of knowledge transfers? In L. Resnick (Ed.), *Knowing Learning, and Instruction: Essays in Honor of Robert Glaser*. Hillsdale, NJ: Erlbaum.

Palincsar, A. & Brown, A. (1984). Reciprocal teaching of comprehension-fostering and comprehension-monitoring activities. *Cognition and Instruction*, 117-176.

Schoenfeld, A.H. (1985). *Mathematical problem solving*. New York: Academic Press.

Singley, M.K. & Anderson, J.R. (1989) *The transfer of cognitive skill*. Cambridge, MA: Harvard University Press.

Singley, M.K., Anderson, J.R., Gevins, J.S., and Hoffman, D. (1989). The algebra word problem tutor. In *Proceedings of the 4th International Conference on AI and Education*. Amsterdam: IOS.

Sweller, J., Mawer, R., & Ward, M. (1983). Development of expertise in mathematical problem solving. *Journal of Experimental Psychology: General*, 112, 639-661.

"Fermat's Last Theorem? I learned about it on Star Trek"

Elliot Soloway
HiCE Project, AI Laboratory
Department of EECS
University of Michigan, Ann Arbor, MI 48109

Abstract: *What* children need to learn in order to have a chance at economic success in our society is changing. In moving from the industrial age into the information age, children must know *how* to learn in order to cope with the changes that will confront them daily. Accompanying this change in what children need to learn, is a change in *how* they learn; as the above real quote illustrates, books are no longer the sole repository of knowledge. In fact, the educational community–and our society as a whole–is now struggling with these issues: the "school restructuring movement" is a significant force in the U.S. today.

In my talk, then, I will attempt a characterization of where education and technology has been and where it is going. In particular, I will attempt to weave together four themes:

(1) the educational goals that we have established and are now questioning (e.g., *accumulation* of knowledge is the goal of current day education),
(2) the "models of the mind" that underlie those goals,
(3) the methods of education that flow from those goals and the models of the mind, and
(4) the types of technological supports for those goals and methods.

In particular, I will argue that the coming generation of computing technology will be the backbone for achieving the goals of the constructivist learning model–a model that is gaining in credibility and popularity.

Cognitive Agoraphobia and Dilettantism: Issues for Reactive Learning Environments

Michael Twidale
Department of Computing
Lancaster University, LA1 4YR, UK

Abstract: This paper reports on a study of students using a basic computer simulation of two frictionless disks colliding in line with perfectly inelastic collisions. The difficulties encountered in discovering a predictive formula are described along with various strategies for managing the task. Although exploratory learning is desirable, it can have certain deleterious side-effects, where a tutorial intervention is desirable. The title refers to two opposite examples of this problem. In the first, the student is so bewildered by the range of options available to her and the size of the task of making sense of the domain that she is unable to act at all. The other case, cognitive dilettantism, occurs when the student moves round the domain at high speed, performing many experiments with little reflection. Productive learning can occur at various points on a continuum between these extremes. The observations carry implications for the development of more supportive learning environments.

Aims of the Study

This study was undertaken as a part of the SIMULATE project funded under the CEC DELTA programme. SIMULATE is concerned with the specification of an Intelligent Simulation Learning Environment (ISLE). Part of Lancaster's contribution was to consider the use of student modelling for simulations. The initial motivation of this study was to supplement a theoretical description of student models for simulations (Self 1990) by providing concrete, realistic examples for the theory. One example student behaviour that had already been used in illustrating the theory was the issue of systematic behaviour; whether the student only changed one variable at a time. We intended to observe the degree to which real students were systematic in order to refine the theory.

At the same time, the study presented the opportunity to observe the use of a simulation by relatively able students to discover the difficulties they encounter. This would enable the composition of a wish list of issues that an ISLE should address.

The Simulation

Rather than use an existing simulation, it was decided to write a simulation that would have certain desirable features. The chief of these was that it should be the most basic kind of simulation that still merited the term. Such a simple simulation should be easier to analyse than more sophisticated ones. Also, any difficulties that students have with such a simulation will transfer to more sophisticated simulations, even if the latter also produce additional difficulties. The simulation developed involves the investigation of two frictionless discs colliding in line with their centres of mass. This could be used in the educational context of teaching about the Law of Conservation of Momentum (LCM). Besides LCM there needs to be a second constraint to uniquely determine the resultant velocities of the discs. The constraint chosen was that the simulation represents perfectly inelastic collisions, in which the resultant velocities must be equal. Thus the formula for the resultant velocity is determined by two equations:

$$\begin{array}{ll} ma.va1 + mb.vb1 = ma.va2 + mb.vb2 & \text{(LCM)} \\ va2 = vb2 & \text{(inelasticity)} \end{array}$$

Where **ma** and **mb** are the masses of discs **a** and **b** respectively, **va1** and **vb1** are their initial velocities and **va2** and **vb2** are their final velocities. Therefore we can derive:

$$va2 = vb2 = (ma.va1 + mb.vb1)/(ma + mb)$$

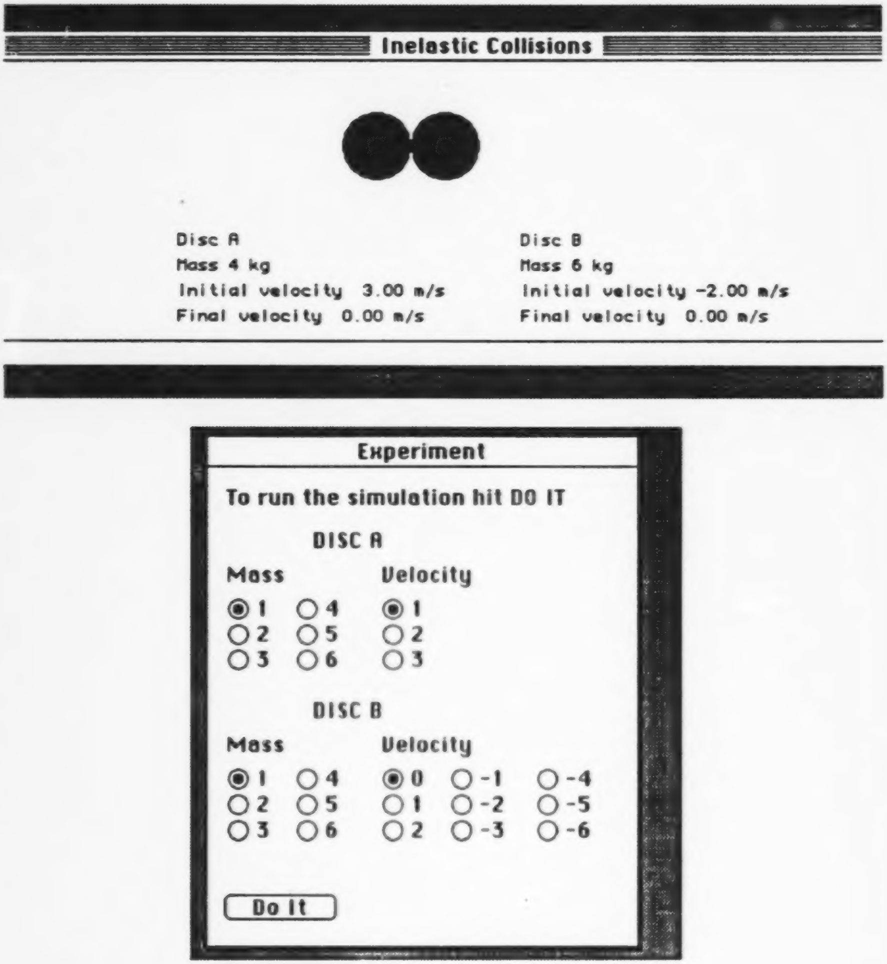
It should be noted that perfectly inelastic collisions, while producing a very simple formula, are somewhat counter-intuitive in that very few real-world collisions are even close approximations of the ideal case. In concrete terms, it is useful to imagine frictionless railway carriages that on colliding always connect and subsequently move together.

Figure 1 illustrates the interface. The simulation was written in Procyon Common Lisp for the Macintosh and was used on a Macintosh with a Portrait screen. The choice of collisions as a domain was also guided by the existence of other work on simulations in this domain (eg. Plotzner et al. 1990 and Interactive Physics, a programme published by Knowledge Revolution).

As can be seen from the figure, the student has four variables that may be altered to one of a preset number of values. The variables are the mass and velocity of two discs, A and B. After setting the variables, the student clicks on "Do It" and the simulation runs, animating the collision and finally printing the resultant velocities. A third window is provided. This is a basic text editor window into which the student may enter comments, record any experimental results etc. This is a minimal note-taking facility and was provided to enable us to observe the extent of note-taking and its nature under minimalist conditions. Other researchers within SIMULATE were investigating the effect of providing more supportive note-taking tools, so this study would provide a useful comparison. Each experiment undertaken by the student is written to a file to aid subsequent analysis.

The Study

A number of undergraduate and postgraduate student volunteers studying a variety of subjects have been observed interacting with the simulation in one hour sessions. The students were told that the aim of the exercise was to test a piece of software and to comment on any things that they found difficult. The students were encouraged both to type into the comments window their ideas and to comment aloud on their thoughts. In particular, they were encouraged to make explicit any predictions they might have about the outcome of an experiment before running it. It should be pointed out that they all appeared to be quite able students. Thus great attention should be paid to those aspects of the simulation that they had difficulty with. By contrast, those elements where they had little difficulty would not necessarily be as straightforward for less able students. So far two kinds of students have been observed: 4 computer scientists and 5 arts/humanities students.



Inelastic Collisions

Disc A
Mass 4 kg
Initial velocity 3.00 m/s
Final velocity 0.00 m/s

Disc B
Mass 6 kg
Initial velocity -2.00 m/s
Final velocity 0.00 m/s

Experiment

To run the simulation hit DO IT

DISC A

Mass
● 1 ○ 4
○ 2 ○ 5
○ 3 ○ 6

Velocity
● 1
○ 2
○ 3

DISC B

Mass
● 1 ○ 4
○ 2 ○ 5
○ 3 ○ 6

Velocity
● 0 ○ -1 ○ -4
○ 1 ○ -2 ○ -5
○ 2 ○ -3 ○ -6

Figure 1. The interface

Nature of the Results

Given the small size of the study, the results must be somewhat anecdotal. This may be justified by the work on situated cognition (Suchman 1987)and by the in-depth studies of learners of Lisp by Anderson et al. (1984) prior to their development of the Lisp Tutor. The analysis of the results attempts to highlight certain activities that learners may undertake when interacting with a simulation which a more supportive learning environment should take account of either by accommodating them or by alleviating the difficulties they engender. The study can make no claims for the proportion of students in a given population that will display these traits. That is the job of a more substantial series of experiments. A companion paper (Twidale 1991) gives the results of the study (including transcripts) in greater detail.

Degrees of Activity

One of the surprising results of the study was the variability in the experimental behaviour of the students. This can be best illustrated by describing two extreme behaviours on a continuum. In the first, which I term cognitive agoraphobia, the student is so bewildered by the range of options available to her and the size of the task of making sense of the domain that she is unable to act at all. This is characterised by sitting passively, staring at the screen, hardly moving at all. The student may be silent or may utter statements such as "I'm really confused" and "I don't know what to do." In its most extreme form this was only observed in one student, who had an arts background which had involved almost no exposure to computers. The opposite extreme of this behaviour may be termed cognitive dilettantism. This can occur when the student becomes so enraptured by the fun of exploring the domain and the effectiveness of the various graphical gimmicks that she moves round the domain at high speed, rather like a child in a science museum.

Alternatively it can occur when the student is very confused but continues rapid activity in the hopes that this will lead to a flash of inspiration, or in order to feel that they are doing something productive. The student with the most extreme version of this behaviour undertook 149 experiments in one hour which compares with an average for the other students of 24 experiments.

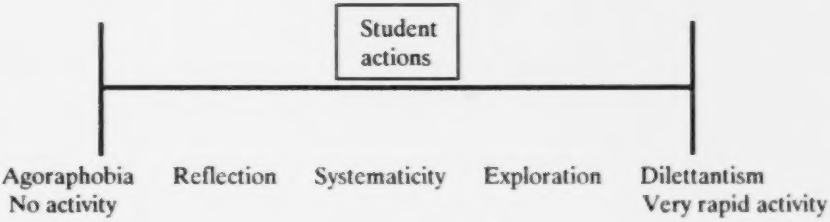


Figure 2. The activity continuum

There are other forms of activity that fit between these extremes as illustrated by Figure 2. Towards the agoraphobia end are the behaviours which may be characterised as reflective learning. Here few experiments are made, the student is mostly silently thinking, perhaps verbalising and sometimes making notes of theories or referring back to earlier notes. In the middle is the systematic behaviour that prior to the study was envisaged as the ideal problem solving behaviour. Here students make steady progress, devising an experiment, making short notes on it (often just recording its result) and then setting up a new experiment with a single variable altered. They are frequently willing to predict the result of the experiment. Beyond this comes more explorative behaviour, involving a more rapid execution of experiments with less systematicity. Students at this point if interrupted and asked what they are thinking about say things like "I'm just seeing what happens." They are much less willing to predict the result of an experiment. Behaviour here can still be a rational choice of exploring the problem space and looking for special features to subsequently investigate in more detail. It can also be associated with qualitative studies (such as the direction of resultant motion) which seem to be able to be undertaken quite rapidly.

The size of the study precludes any useful figures on the percentages of people fitting at certain points on the above scale. Indeed it appears that people move around on the scale at different occasions. One particularly productive cycle that was observed is to begin at the exploratory mode, look for a special case (such as a resultant velocity of 0), move to investigating it systematically, reflect on the results, propose and test a formula and then cycle back to exploratory mode to find another sub-case.

Student-Directed Learning is Unpredictable

The provision of a supportive learning environment such as a simulation is often extolled as preferable to a traditional tutoring system (even an intelligent one) because it enables the learner to take greater control of her learning. We have already seen that this can lead to problems if the learner is unable to take control. Even when the learner does take control there may be unexpected results. The learner may set her own learning agenda that is different to that intended by the developer. An example from the study was where a student undertook an investigation solely in qualitative terms. That is, he decided to investigate the circumstances under which a collision led to the discs stopping dead, and also to investigate under what conditions they had the same final velocity (in fact, they *always* have the same final velocity). Once he had produced a description of the circumstances under which these occurred he declared "I think that's all I can find. I can't think of anything else to look for," and stopped. A different learning agenda is a problem for two reasons. Firstly, the system developer may have particular learning objectives for the student which fail to be realised because the student happens to set herself a different goal. This may be a deliberate decision on the part of the learner, but it may be that she was not even aware that the intended learning goal existed. Secondly, an ISLE which attempts plan recognition in order to offer more appropriate advice needs to be able to take account of alternative goals otherwise it may severely misinterpret the student's actions.

Qualitative Reasoning

Most students did some form of qualitative reasoning and appeared to get an intuitive feel (including making correct predictions) before proceeding to develop a quantitative formula. For example they would investigate the circumstances under which the resultant velocity was positive, zero or negative. They also referred to trying experiments when a disc was "heavier" or "moving faster." It would seem to be useful strategy to investigate the domain qualitatively before moving on to attempting a quantitative description. It is interesting that this spontaneous activity was so similar to that advocated by White and Fredericksen (1990).

Finding the Formula

Only 3 of the 9 students were able to generate a formula that accurately described the resultant velocity in all cases. The others were only able to find formulae for special cases. This was despite considerable efforts on the part of the students who not only appeared to be articulate and intelligent but very motivated. This difficulty is interesting given how simple the simulation formula is. It is worth comparing the performance with that of SMITHTOWN (Shute and Glaser 1990) which teaches about very many economic concepts that have the potential to interact in confusing ways. Despite this the users of SMITHTOWN learn quite a lot about economics. One reason for this difference must be that SMITHTOWN offers a far better supportive learning environment. However I believe another reason to be the nature of what the student was trying to discover. In SMITHTOWN it was a group of qualitative descriptions of the effects of changes of economic variables. In our simulation the student was also attempting to create a formula. It appears that the latter is a considerably harder task. Thus in developing a simulation environment, if one wants the student to find a formula, one is likely to need to provide considerably more support than if the student only needs to discover a qualitative relationship.

It is also worth noting that just because a student obtains a formula, this does not necessarily imply that she has the same understanding of the formula as the simulation developer has and wishes to impart. At first glance at the collisions simulation, it appears that it would be a useful device to help learn the Law of Conservation of Momentum. Following from that, any student who could consistently predict the result of an experiment might be deemed to have necessarily remembered and to be using, or to have induced that law (in the latter case she would have the formula but not of course the name). However this was found not to be the case. After each study the students were asked about the Law of Conservation of Momentum and with one exception, none of them realised its relevance (or even appeared to recall it) while using the simulation. (The exception, while not explicitly referring to it or using it, appeared to implicitly use it by his choice of formula fragments.)

The case of subject IH illustrates the lack of understanding. Even though he was able to produce a formula that could give the correct final velocity in all cases, he never 'discovered' the Law of Conservation of Momentum (LCM). His formula was:

$$va2 = (ma.(va1-vb1) / (ma+mb)) + vb1 \qquad \{1\}$$

The formula derived from the Law of Conservation of Momentum earlier is:

$$va2 = ma.va1+mb.vb1 / (ma+mb) \qquad \{2\}$$

{1} and {2} are algebraically identical but may be perceived quite differently by students.

In the debriefing, IH admitted to having studied LCM at school but he only realised the connection with the simulation when it was pointed out to him. Thus the knowledge of LCM was not explicitly available to him in his use of the simulation. This would imply that the law is not one that is well remembered, possibly because in a friction-full world it is counter-intuitive.

Formula {1} may have very little 'meaning' attached to it. It could be regarded as merely an arbitrary formula that successfully predicts the discs' activity. It can remain devoid of meaning even though the student has derived it himself by experimentation. However, IH did perform qualitative reasoning to get to this formula. This could be used to add some meaning, but it is debatable how much IH could describe the understanding obtained.

Difficulties of Plan Recognition

As the descriptions earlier imply, there are a number of strategies that a student may employ to help them to learn with a simulation. As well as organisation and control, they cover issues such as error detection and recovery. These are described in more detail elsewhere (Twidale 1991). Before the study was undertaken, the main control strategy had been believed to be systematicity: varying a single variable at a time by small amounts. However other activities that are rational but do not fit the simple systematicity criteria were observed. One example is the consistent changing of more than one variable so as to maintain a certain ratio. For example the student may double the mass of disc a while halving its velocity. Also the student may perform quite erratic experiments, just to get a feel for the kinds of results the simulation produces. Even students who were at other times systematic did this. This exploration phase could occur at various times:

1) On first using the system.
2) After investigating a special case and when looking for another case to investigate.
3) In order to test a formula with 'random' values.

Thus even attempting to recognise systematicity is not as straightforward as might be expected.

It is interesting that the kinds of student behaviour observed in using the simulation are similar to those observed in a quite different domain; a tutoring system for the learning of techniques of Logic proofs (Twidale 1990). For example in that study it was observed that a student may deliberately perform an action that she believes to be incorrect because she knows that the system will interrupt and offer a suitable explanation. This is done because although she feels the action is incorrect, she does not know why it is.

Research Issues

The study raises issues that it would be productive for a larger and more systematic set of experiments to investigate more thoroughly. These include:

- Whether the agoraphobia-dilettante continuum is a useful description of activity which can be utilised by an ISLE.
- The proportions of various populations at the extremes who are consequently unable to learn much from simulations.
- The effect of help, advice and more supportive interfaces in moving these people towards the middle of the continuum.
- The degree to which people move about the continuum during an interaction.
- The effect of experience of computing debugging skills on learning using simulations. (This appeared to have a positive effect on learning which is at variance to the findings of the SMITHTOWN study on the transfer of science expertise.)

Conclusion

The study has revealed some of the complexities involved in learning with a reactive environment. Part of the complexity is inherent in any learning process and it is only when the requirements of developing an intelligent learning environment necessitate an in-depth study of the behaviour of a few learners that the complexities become apparent. In this sense, the study is very similar to Anderson's (1984) study of learners of Lisp. Also a reactive learning environment adds extra complexity as it offers the user many additional learning opportunities that lead to user behaviours that are different from those possible in traditional pen and paper environments. Such activities make plan recognition and student modelling even more challenging.

References

Anderson J.R., Farrell, R. and Sauers, R. (1984). Learning to programme in Lisp. *Cognitive Science* 8, 87-129.

Plotzner, R., Spada, H., Stumpf, M. and Opwis, K. (1990). Learning qualitative and quantitative reasoning in a microworld for elastic impacts. Research Report, Psychological Institute, University of Freiburg. To be published in the *European Journal of Psychology of Education*.

Self, J. (1990). Tools for building learner models. Proceedings *Delta and Beyond Conference*, The Hague.

Shute, V.J. and Glaser, R. (1990) A large-scale evaluation of an intelligent discovery world. *Interactive Learning Environments* 1, 51-77.

Suchman, L.A. (1987). *Plans and situated actions*. Cambridge University Press.

Twidale, M.B. (1990). Student activity in an intelligent learning environment. Lancaster University Computing Department AI Group Report No. 62.

Twidale, M.B. (1991). An investigation into the use by students of a basic computer simulation. Lancaster University Computing Department AI Group Report No. 63.

White, B.Y. and Frederiksen, J.R. (1990). Causal model progressions as a foundation for intelligent learning environments. *Artificial intelligence* 42, 99-157.

An Experimental Evaluation of an Intelligent Tutor for Diagnostic Problem Solving

Vijay Vasandani and T. Govindaraj
Center for Human-Machine Systems Research
School of Industrial and Systems Engineering
Georgia Institute of Technology, Atlanta, Georgia 30332

Abstract: This paper focuses on the experimental evaluation of an intelligent tutor designed to train operators to troubleshoot a simulated marine power plant. Three training conditions were explored: (a) training on simulator alone; (b) training with the aid of a passive tutor; and (c) training with the aid of an active tutor. Following the training sessions, the performance of subjects in all three conditions was analyzed in identical data collecting sessions.

The experimental results reveal that the simulator alone as a training tool is inadequate. However, excessive aiding during training leads to the trainee becoming somewhat dependent on the tutor. The training condition also appears to influence the diagnostic strategies of the trainees. Those trained on the simulator alone rely heavily on educated guesses and unguided search for the problem. On the other hand, those trained with the tutor develop efficient troubleshooting skills.

Background

Use of intelligent tutoring systems (ITS) can be economical for training supervisory control operators of complex dynamic systems. Such tutoring systems should help organize system knowledge and operational information and also provide practice to develop appropriate skills. We have developed and implemented such an ITS on an Apple Macintosh II computer for the marine power plant domain. The ITS comprises of **Turbinia**, a simulated power plant; **Vyasa**, the tutor; and mouse-based direct manipulation graphical interfaces. The tutor, **Vyasa** functions in two modes: passive and active. In the passive mode, it provides help upon request and without intervention. In the active mode, it provides assistance with intervention. Together, **Turbinia** and **Vyasa** can be used to train operators to troubleshoot a marine power plant. Architectural and implementation details of this instructional system are provided in Vasandani et al. (1989 and 1990) and Vasandani (1991). In this paper we describe the experimental evaluation of **Turbinia-Vyasa**.

Experimental Evaluation of Turbinia-Vyasa

Prior to beginning formal experiments to test the effectiveness of **Turbinia-Vyasa**, the tutor was evaluated with the help of a naval officer. This evaluation was informal and somewhat subjective. Its primary goal was to ensure that the instructional material presented to the student was technically correct, properly stated, and consistent with the current training program for engineers in the U.S. Navy.

After the subjective evaluation, a pilot experiment was conducted. The purpose of the pilot experiment was to validate the instructional material and to determine software errors. Four graduate students from Georgia Institute of Technology participated in the pilot study. The pilot study identified several discrepancies in the instructional manual and errors in software. These were promptly corrected. Formal experiments began after this preliminary evaluation.

Formal Experiments

In the formal experiments, performance of subjects trained with and without the tutor was compared. There were two goals in the experiment: (1) determining the effectiveness of the tutoring architecture and methods for knowledge representation and (2) establishing the usefulness of computer-based training programs over traditional means of training operators to troubleshoot complex dynamic systems.

The experiment consisted of two phases: training and data collection. In the training phase, subjects were exposed to one of the three instructional methods: (a) training on simulator alone; (b) training with the aid of a passive tutor; and (c) training with the aid of an active tutor. During data collection, trained subjects from all three conditions attempted to solve the same set of problems unaided by the tutor.

Experimental Design

There were two main factors of interest in the experiment: training condition and seen status of the problem. There were three training conditions. The first was a baseline condition where training was provided using just the simulator **Turbinia**. The second and the third training conditions used the computer-based tutor **Vyasa**.

The second factor of interest was the seen status of the problem. Seen status had three levels: seen once, seen twice and unseen. These levels indicated the number of times the problem was seen by the student during training.

Subjects (nested within training condition) and problems (nested within seen status) were two other factors that could account for variations in the experimental data. These along with the main factors and their interactions make a complete list of sources of variation considered in this experiment.

Training condition and seen status were fixed factors, and subjects (nested within training condition) and problems (nested within seen status) were random factors. Therefore, a mixed model was used to analyze data from the experiment.

Equipment and Materials

Turbinia (simulator) and **Vyasa** (tutor) were installed on a dual screen Apple Macintosh II workstation. Instructions for the three experimental conditions were audio-taped in a female voice to be played back to the subjects during their first training session. In addition, three instructional manuals, one each for the three experimental conditions, were prepared for distribution to the subjects prior to the experiment.

Subjects

Thirty cadets from the Georgia Institute of Technology Naval ROTC unit participated as subjects. All subjects had a basic understanding of the theory of marine power plants. Each subject was paid $6 per session for every session completed. In addition, an award of $25 was promised for the best troubleshooter in each of the three groups based on performance in the two data collection sessions.

Prior to the start of the experiment, the subjects were randomly assigned to three groups of ten each. Group I was trained using just **Turbinia**, the simulator. Subjects in Groups II and III were aided by the computer-based tutor **Vyasa**. While **Vyasa** functioned in the passive mode for Group II, it functioned in the active mode for Group III.

Training Phase

The first 10 sessions, for each group, were used for training of subjects. Each session lasted 45 minutes. The sessions were run on consecutive days with typically one session per day.

The first training session for each group introduced the system using a single problem. Audio taped instructions, different for each group, were used during this session. These instructions introduced the subjects to the interface and valid forms of interactions. The capabilities of the passive and active tutor were also demonstrated to the groups using them.

After the first session, subsequent training sessions had three problems each. A subject had a maximum of 13 minutes to solve each problem. If the subject solved the problem early, the next problem was immediately presented. Thus, if the subject solved one or more of the three problems in a session within the allotted time for each problem, the session was completed in less than 45 minutes.

At the end of each problem the subject was provided the solution. While solutions presented to subjects in the tutor condition were accompanied by an explanation, no such explanation was provided to subjects using the simulator alone.

Data Collection Phase

The data collection phase consisted of two sessions. These sessions were run on consecutive days immediately following the completion of training. During the data collection sessions, the subjects interacted with the simulator only, unaided by any tutor, irrespective of their training condition.

Each data collection session was approximately 50 minutes long and consisted of 5 problems. If the subject solved the problem within the 10 minute period allocated for each problem, the next problem was presented immediately. However, unlike the training sessions, no solution was provided to the student at the end of each problem.

Training and Test Problems

Twenty-four problems were identified for use in the experiment. Since we wished to study the effect of training for familiar as well as novel situations, 5 of these 24 problems were kept aside for the data collection phase. The remaining 19 problems were available for use in training.The training phase comprised of 1 single-problem session and 9 three-problem sessions. In these sessions, 9 problems were shown twice to the subjects. Selection of these

9 problems was done randomly. However, the same problem was never presented the second time within a span of 3 consecutive sessions.

In the test sessions, in addition to the 5 unseen problems, subjects were given 5 problems from among those seen during training. Three of these problems appeared twice during training and two were seen only once.

The order in which problems were presented during training and test was identical for all groups. In the test sessions, seen and unseen problems were alternated beginning with an unseen problem. The actual order in which the problems appeared was, however, determined randomly.

Performance Measures

Although the ultimate goal in troubleshooting is to successfully identify the failed component, there were several other performance measures considered in the experiment. This section describes the measures used to evaluate the troubleshooter's performance.

Number of problems solved. Successful diagnosis of a fault is an important measure of troubleshooting ability. A problem is considered solved if a successful diagnosis is made within the time (10 minutes) allocated for each problem.

Troubleshooting time. For those subjects who are successful in solving the problems, the total amount of time taken for solution is a valuable measure of their performance. Those who take less time are considered better troubleshooters.

Number of informative actions. Actions that are taken to obtain gauge readings are informative because they help the students access the system state information needed to solve the problems. The total number of such informative actions taken to solve the problem provides a measure of the student's overall troubleshooting ability. A smaller number indicates a better diagnostic performance.

Percentage of relevant informative actions. Even though every informative action has some information content, only some are directly relevant to the failure. Since the total number of relevant informative actions necessary to solve a problem depends upon the problem, the percentage of relevant informative actions is a good measure of performance.

Nature of diagnosis. When a student correctly solves a problem but has not gathered sufficient evidence to do so, the diagnosis is considered premature. On the other hand, when the student is unable to diagnose the fault even after sufficient evidence implicating the failed component has been gathered, the diagnosis is considered overdue. But, when the student integrates diagnostic information properly, the diagnosis is neither premature or overdue. Instead, it is considered timely. Categorizing diagnoses as premature, timely or overdue provides a subjective measure of diagnostic performance.

Percentage of guesses. An incorrect diagnosis that implicates a component that could not have failed, based on observed symptoms, is a consequence of pure guesswork or inaccurate troubleshooting knowledge. Such an incorrect diagnosis is considered a guess and fewer guesses indicate better troubleshooting performance. Since the number of incorrect diagnoses, in terms of guesses and probable hypotheses, may depend on the problem, percentage of diagnosis for each problem that were guesses is a reasonable measure of troubleshooting performance.

Number of unaffected schematics/subsystems/fluid-paths investigated.. Affected schematics, subsystems and fluid-paths are those that have gauges with abnormal readings. Investigating components in schematics, subsystems or fluid-paths other than the ones affected by the failure reflects the student's inability to relate symptoms to the structural and functional location of the power plant. Thus, the number of such investigations is also a measure of performance.

Prior to conducting the experiments, we expected that those trained on the simulator alone would perhaps develop an unguided search strategy to locate the fault. Therefore, we expected them to make more guesses, take a lower percentage informative actions that are relevant, make more premature diagnoses and investigate more unaffected schematics, subsystems and fluid-paths. Since they would rarely be using concrete reasoning, we also expected them to solve the problem in less time but with many attempts of incorrect diagnosis.

Experimental Results

Results from the analysis of the experimental data to compare the performance of the subjects from each of the three training groups are described in this section. This analysis was conducted using the SAS General Linear Model (GLM) and using Type III sum of squares. Even though the experimental design was fairly well balanced, some of the sum of the squares generated were not independent and hence some of the test statistics had to be computed using Satterthwaite's approximation technique (Milliken and Johnson 1984). The results from the analysis of variance are shown in Table 1. A brief discussion of these results for each performance measure is presented next.

(1) **Number of problems solved.** Subjects in Group II solved the maximum number of problems followed by subjects in Groups I and III. In addition, each group solved more seen problems than unseen problems. However, the effects of both training condition and seen status were statistically insignificant. Relatively poor performance of subjects in Group III can be attributed to three factors: (i) a single subject was unable to solve five problems; (ii) subjects were reluctant to guess the failures; and (iii) some subjects became dependent on the tutor.
(2) **Troubleshooting time.** The subjects in Group I were the quickest to solve the problems while those in Group III were the slowest. This is not at all surprising because the unaided group did not have a guided strategy to solve the problems and relied heavily on guessing. Guessing as opposed to concrete reasoning takes less time. Also, subjects in all the three groups were quicker to solve seen problems than unseen problems.
(3) **Number of informative actions.** The subjects in Group I were unable to utilize the information as effectively as subjects in the other two groups and therefore had to conduct more diagnostic tests to solve the problem. Also, the number of informative actions taken were higher for unseen problems in each group. However, the effects of both training condition and seen status were statistically insignificant.
(4) **Percentage of relevant informative actions.** The percentage of the informative actions that were relevant to the failure being investigated was significantly smaller for Group I. This suggests that those trained by the tutor were better able to identify and conduct diagnostic tests that were useful for solving the problem. However, in each group, the effect of seen status on the percentage of relevant informative actions was insignificant even though the number of relevant informative tests performed was lower for unseen problems.

Table 1

Results from Analysis of Variance

Performance Measures	Main Factor			
	Training Cond	Seen Status	Subject (Cond)	Problem (Seen)
Problems solved	NS	NS	NS	S
Troubleshooting time	S	S	S	S
Number of informative actions	NS	NS	S	S
Percentage of relevant informative actions	S	NS	S	S
Nature of diagnosis	S	S	NS	S
Percentage of guesses	S	S	S	S
Number of unaffected schematics investigated	NS	S	NS	NS
Number of unaffected subsystems investigated	S	NS	S	S
Number of unaffected fluid-paths investigated	S	NS	S	S

S: Effect significant at the level of 0.01

NS: Effect insignificant at the level of 0.01

(5) **Nature of diagnosis.** The effects of both the training condition and the seen status of the problem had a significant influence on the nature of the diagnosis. Group I provided more instances of premature diagnoses as compared to Groups II and III. Furthermore, instances of premature diagnoses were lower with seen problems for all the three groups. Also, for Group I, only 50% of the problems were solved in a timely manner as compared to 77% and 75% for Group II and III respectively. Finally, for Group I, many more diagnoses were overdue as compared to the two aided groups. This suggests that the subjects in Group I were not as good at integrating diagnostic information as those in the two aided groups.

(6) **Percentage of guesses.** The percentage of incorrect diagnoses that were guesses was significantly higher for Group I. For Group I, 68.8% of the incorrect diagnoses were guesses as compared to 33.49% for Group II and 29.38% for Group III. Also, for Group I, evidence of guessing strategy was noticed in 60% of the problems whereas for Groups II and III it

was noticed in 39% and 30% of the problems respectively. In addition, the subjects in Group I often used guessing as a primary strategy whereas the subjects in Groups II and III started guessing only when they were running out of time. For all the three groups, however, guessing was more prominent with unseen than with seen problems.

(7) **Number of unaffected schematics/subsystems/fluid-paths investigated.** Group I had several more instances of investigations in unaffected schematics than the two aided groups, although the difference was statistically insignificant. However, instances of investigations in unaffected subsystems and fluid-paths were significantly higher for subjects in Group I. The effect of seen status was also significant for investigations in unaffected schematics. More unaffected schematics were investigated when solving the unseen problems.

The effects of individual differences among the subjects within the group, and the problem being investigated were also responsible for a lot of variation in the data for each performance measure.

Conclusions

From an analysis of the results, it is clear that the tutor in both the passive and the active modes helped the students to develop useful troubleshooting strategies. Those trained by the tutor formed plausible failure hypotheses based on observed symptoms and systematically eliminated them by conducting appropriate diagnostic tests. In comparison, those trained without the tutor did not develop good troubleshooting strategies. They relied rather heavily on guessing the solution. Furthermore, the tutor helped the students to recognize and integrate crucial diagnostic information in a timely manner that the students without the tutor were unable to do.

The data also indicated that the effectiveness of a tutoring strategy depended upon the individual student. For example, the strategy of providing explanations for all observed symptoms for each problem was intended to help the students develop a proper causal model of fault propagation. Some students who learned to map salient symptoms to causes from these explanations became overly conservative. During troubleshooting they spent a lot of time eliminating all probable hypotheses linked to an observed symptom even when sufficient evidence in support of a highly probable hypothesis had been collected. Another tutoring strategy adopted by the active tutor was to provide help in building, refining, and eliminating failure hypotheses. In this capacity the active tutor came to be perceived by some students as an on-line associate. These students often took the help of the active tutor to refine their failure hypotheses and thus became dependent on the tutor to solve problems. Performance of these students deteriorated when the active tutor was withdrawn.

In summary, results of the experiment show that a simulator alone is inadequate. A simulator in conjunction with a computer-based tutor can help develop efficient troubleshooting skills. However, not all students are equally receptive to every tutoring strategy and provisions must be made in training programs for individual preferences and differences in abilities and styles.

This research was supported by contract N00014-87-K-0482 from the Manpower R&D Program, Office of Naval Research (Dr. Susan Chipman, contract monitor). We wish to thank the thirty volunteer cadets for participating as subjects in our experiment. Dr. Christine Mitchell at Georgia Tech also deserves our thanks for her help with many design and implementation issues and with statistics.

References

Milliken, G.A., & Johnson, D.E. (1984). *Analysis of messy data volume I: Designed experiments*. New York: Van Nostrand Reinhold Company.

Vasandani, V., & Govindaraj, T. (1989). An intelligent tutor for diagnostic problem solving in complex dynamic systems. *Proceedings of the 1989 International Conference on Systems, Man, and Cybernetics* Boston: MA, pp. 772-777.

Vasandani, V., & Govindaraj, T. (1990). Knowledge representation and human-computer interaction in an intelligent tutor for diagnostic problem solving. *Proceedings of the 1990 International Conference on Systems, Man, and Cybernetics* Los Angeles: CA, pp. 665-667.

Vasandani, V. (1991). *Intelligent tutoring for diagnostic problem solving in complex dynamic systems*. Doctoral dissertation in progress, Georgia Institute of Technology.

How Many Modes of Cognitive Operation? Some Evidence from a Brain Monitoring Study of Quantification Processes

J.G. Wallace
Information Technology Institute
Swinburne Limited, John Street, Hawthorn, Vic. 3121, Australia

R. Silberstein
Centre for Applied Neurosciences
Swinburne Limited, John Street, Hawthorn, Vic. 3121, Australia

K. Bluff
Information Technology Institute
Swinburne Limited, John Street, Hawthorn, Vic. 3121, Australia

Abstract: In seeking to improve teaching and learning one of the most significant issues centres on the range and nature of modes of cognitive operation. Performance on a series of cognitive tasks which appear to exemplify distinctions in modes of operation is being explored by a combination of cognitive modelling, based on a hybrid, sub-symbolic/symbolic approach to representation, and brain monitoring of the association of event related potentials (ERPs) with cognitive processing. Results obtained in studying quantification processes suggest that the sub-symbolic, primary conscious operations involved in subitizing give rise to distinctive ERP profiles. More temporally extended tasks such as counting by ones require an alternative approach to mode identification. The feasibility of steady state evoked potential methodology fulfilling this requirement is being explored.

Introduction

Exploration of the fundamental nature of cognitive processes continues to be a source of problems and ideas for those seeking to improve teaching and learning. One of the most significant issues centres on the range and nature of modes of cognitive operation. As Norman (1986) has pointed out "People do seem to have at least two modes of operation, one rapid, efficient, subconscious, the other slow, serial and conscious." This issue is being researched in different contexts such as the distinction between implicit and explicit modes of learning (Reber, Kassin, Lewis & Cantor, 1980; Berry and Broadbent, 1988), unconscious metaprocesses and conscious access in cognitive development (Karmiloff-Smith, 1986), and primary and secondary consciousness in brain function (Edelman, 1989). The general aim of the research to be reported is to tackle the issue by capitalizing on two methodological trends which appear likely to shape much of the investigation of cognitive processes in the 1990's. These comprise a heavy reliance on sub-symbolic representation in cognitive modelling and the acquisition of empirical data by the monitoring of brain function. The search for relatively brief and more extended tasks suitable for the investigation of the extent to which they rely on sub-symbolic and symbolic processing has, to date, centred on the topics of quantification and implicit and explicit knowledge and modes of learning.

Quantification

Quantification processes provide the basis for the representation of quantity and the semantic substrate for the mapping of number names. An analysis by Klahr and Wallace (1976) defined the major quantification processes as subitizing, counting (subdivided into 'counting by ones' and 'subitizing and adding') and estimation. Subitizing and counting are examples of relatively brief tasks that appear to tap sub-symbolic and symbolic processing to varying degrees.

In almost every early study of quantification of collections of n items (usually random dot patterns) where $1 \leq n \leq 30$, striking discontinuities occur in the region of $n = 6 \pm 1$. This is true for plots of reaction time versus n, error rate versus n and confidence versus n. These discontinuities define subitizing as the process that produces the results up to the point of discontinuity. Beyond that point other processes are in operation.

The nature of the processes underlying subitizing remains problematic. The early studies referred to it as 'immediate apprehension' and attempted to establish the range for which the time to quantify a collection is independent of the value of n. Further work, summarised in Klahr and Wallace (1976), has revealed an increase in response latency with n and supports the inclusion of sequential processes in subitizing. Analysis of the latency data suggests that the increase with n is linear but the small number of data points being fitted prevents this from being a firm conclusion.

Starkey et al. (1980) on the basis of their studies of quantification in infants, conclude that "infants possess a primitive form of non-verbal counting" that "may underlie (some) subitizing phenomena" and that might "share some component processes with verbal counting." Whether 'nonverbal counting' is an appropriate label for the processes involved in subitizing remains a moot point.

The numerical upper limit of subitizing remains unclear in spite of considerable chronometric analysis of latency data. The discontinuity at 6 ± 1 detected in the early studies has been replaced by an upper limit at 3 or 4. The possibility remains, however, that the subitizing limit may be 4 or 5 and is concealed by subjects subitizing on some trials and counting on others in this part of the numerical range.

Subitizing seems to have predominantly sub-symbolic characteristics. Counting by ones appears to have a greater symbolic emphasis while subitizing and adding evinces both sub-symbolic and symbolic features. Means of identifying and discriminating processes in terms of their reliance on symbolic and sub-symbolic operations are required. Rumelhart et al. (1986) in discussing the relationship between PDP models and more conventional sequential systems indicate their belief that processes that happen very quickly, in 250 to 500 milliseconds, occur essentially in parallel and should be described in terms of parallel models. Processes that take longer, they believe, have a serial component and can more readily be described in terms of sequential information processing models expressed, for example, in the form of productions.

The general limitations of latency data and chronometric analysis in achieving the objective of process identification and discrimination in the area of quantification have already been outlined. An approach such as Rumelhart's 'rule of thumb' for discriminating sub-symbolic and symbolic operations is much too coarse to make a major contribution to analysis. Data that offer the potential of more fine-grained analysis are required.

Initial work focussed on validation of a range of observed and hypothesized differences between the processes of subitizing and counting by ones.

(1) Subjects report no conscious awareness of the subitizing process prior to generation of a number label for a collection. The process, also, appears to occur within a single visual fixation. Counting by ones, in contrast, produces awareness of an iterative process of object localisation and number label generation involving a systematic attention directing technology based on eye movements sometimes accompanied by the physical movement of elements in the target collections.

(2) Latency data from previous studies (Klahr and Wallace, 1976), indicate an intercept in the subitizing range (1-3) of 495 msec and an increase with n of 46 msec. The process employed in the 4-10 range produces an increase with n of 307 msec and a negative intercept consistent with the operation of processes other than subitizing. No latency data distinguishing the operation of counting by ones from subitizing and adding in the 4-10 range or covering the use of counting by ones in the 1-3 range are available.

(3) The reaction times for subitizing lie in the 540-630 msec range. This places it in the latency bandwidth regarded as indicating dependence on sub-symbolic processes. Any confirmation of a latency increase with n, however, raises the issue of its source in the operation of parallel, sub-symbolic processes.

A converging strategy has been adopted in investigating these differences in which the constraints and cross-checks emerge from the interaction of empirical data derived from cognitive psychological experiment, results derived by brain monitoring during cognitive performance and the sufficiency test provided by adoption of cognitive modelling as a theorizing medium. Although the current emphasis is on the acquisition of empirical data and brain monitoring, the groundwork for cognitive modelling has been laid. The orientation adopted reflects Clark's (1989) view that claims for the sufficiency of either symbolic or sub-symbolic approaches to exclusively account for cognitive functioning should be rejected. It is assumed that a full spectrum of relations exists between symbolic and sub-symbolic representation and between the two types of cognitive modelling based upon them. The analysis of each cognitive task is undertaken as potentially requiring a hybrid approach to do justice to its qualitatively 'mixed' nature.

In the case of quantification the strategy involves developing specific models of the two methods in terms of their apparent dependence on sub-symbolic and symbolic operations. The overlaps and interrelationships of the specific models are being interpreted and resolved within the framework of a general model of cognition developed by Wallace and Bluff (1989).

The main vehicle for modelling is the Rochester Connectionist Simulator (Goddard et al., 1989). This is a software shell designed to provide a maximally flexible environment for the construction of sub-symbolic models in the form of connectionist or neural networks. It, also, permits straightforward interfacing of networks with other modules created in C code. Modelling of symbolic processes is carried out in C code and this ease of interfacing is utilized to construct general and specific models in hybrid, symbolic/sub-symbolic form.

Preliminary exploration of connectionist learning mechanisms is underway to assess their suitability as substitutes or complements for the symbolic learning mechanisms already incorporated in the general model of cognition. The potential of a general learning unit (GLU) based on an extension of the work of Barto (1985) and of an associative learning mechanism modelling the single neuron level (Rogers, 1991) is being investigated.

Brain Monitoring and Event Related Potentials

The monitoring of brain electrical activity by means of scalp electrodes has a long history. More recently, numerous studies have examined the effects of cognition on brain electrical activity evoked by sensory stimuli (Gevins and Cutillo, 1986). These responses, known as event related potentials (ERPs) are often less than 5 microvolt in amplitude and have a duration of 0.4 to 0.8 seconds (McGillen and Aunon, 1987). The current study is breaking new ground by exploring the association of ERPs with cognitive processing during a variety of tasks which highlight apparent differences in mode of cognitive operation.

In considering the association of ERPs with cognitive operations in general and quantification processes in particular the framework adopted reflects the fundamentals of Edelman's (1989) account of primary and secondary consciousness in brain function. Primary consciousness is defined as the result of the ongoing discrimination of present perceptual categorizations by a value-dominated, self/non-self memory. This memory contains a record of perceptual categorizations, derived from previous experience, arranged as concepts and linked to values derived from the phenotype provided by the evolutionary emergence of the brain stem.

The beginnings of higher order consciousness are seen in an emerging consciousness of the distinction between categories related to the 'self' and those related to entities classifiable as 'non-self.' This provides the earliest basis for subject-predicate relations. With this ability, the beginnings of higher order consciousness are in place. But only with the appearance of true language in a social context can this form of consciousness fully flourish. Because human beings have higher order consciousness, they cannot subjectively experience or reconstruct primary consciousness without the intrusion of some additional components of direct awareness and higher order consciousness.

Each of the selected areas of cognitive operations involve task performances that appear to be heavily dependent on sub-symbolic or symbolic processes. Those with a high degree of dependence on sub-symbolic processes, also, exhibit characteristics consistent with the involvement of primary consciousness. Consequently, the involvement of sub-symbolic processes is difficult to detect and to discriminate from the intrusive effects of symbolic processing and higher order consciousness. In seeking to use brain monitoring techniques in tackling these problems, the specific hypotheses to be tested are based on Edelman's descriptions of the distinctive features of the neural events underlying the operations of primary consciousness and higher order/secondary consciousness.

Justice cannot be done to the richness of Edelman's account since the brain monitoring techniques available are confined to data acquisition from cortical areas. However, a broad range of cortical sites are being monitored since the specific hypotheses to be tested reflect Edelman's proposals on the nature of the relationship between primary and secondary consciousness at the cortical level. This relationship is generally defined in terms of the pre-frontal, cingulate, parietal, temporal, sensory and motor cortex and, more specifically, Broca and Wernicke's areas.

The quantification data presented above are consistent with Edelman's view on latencies produced by perceptual processes mediated by primary consciousness. From this perspective subitizing becomes an aspect of the 'values derived from the phenotype.' It is viewed as a phylogenetic result of the survival value inherent in a basic and limited ability to partition sensori-perceptual input from the environment into the type of global segments conforming to the principles of cardinal numerosity. More specifically, this leads to the prediction that the subitizing process will be associated with the recording of relatively high ERPs in the pre-frontal cortex designated by Edelman as centrally involved in the primary conscious process of matching current perceptual input with the memory record of perceptual categorization derived from previous experience.

In contrast, the data on counting by ones in the 4-10 range suggests a conscious process, relatively extended in time and involving a sequence of operations including regular updating of verbal symbols. Prediction of the distribution and relations of the ERPs produced by counting by ones is difficult. It would be expected, however, that relatively high levels of activity would be recorded in cortical areas, such as Broca's area, which are associated with speech production.

Experimental Results on Quantification

Data gathering on quantification takes place in a single experimental situation operating in two conditions designed to discover how to identify and discriminate the use of subitizing and counting by ones. The main features of the experimental situation are identical to those validated for the purpose and used by Klahr and Wallace (1976).

Each subject undergoes both experimental conditions. Subjects are individually presented with patterns of 1 to 10 dots randomly generated on a standard video monitor controlled by a computer. Their responses trigger a voice-actuated relay which is sensed by the computer, yielding latency measurements accurate to the nearest millisecond, from the time the pattern appears on the screen until the relay is activated. No indication of the range of numbers represented by the displays is provided before or during their presentation. This is intended to minimise expectancy and 'end of range' effects.

Eye movements are monitored during all trials and, where appropriate, only those ERPs uncontaminated by eye movements are averaged. In both experimental conditions dot collections in the range 1-10 are presented. In the first condition no instructions on the processes to be employed in quantifying the collections are given to the subjects. They are instructed to quantify the collections as rapidly as they can. In the second condition explicit instructions to use counting by ones as the only method of quantification are given.

The data gathered on subjects' performance comprises accuracy information, response latencies and ERP data. The 30 recording locations of ERP data include all the scalp positions in the international 10 - 20 system with additional sites located midway between the 10-20 locations. All electrodes are held in position using a specifically designed helmet (Ciorciari et el., 1987). The occurrence of the onset of the visual display and the subject's response are flagged in the ERP records to facilitate analysis and interpretation of the results.

The reaction time results in the speeded experimental condition reveal the features associated with subitizing. The familiar discontinuity exists between 3 and 4 and the specific latencies and rate of increase with n obtained in earlier studies are reproduced. Counting by ones when applied to collections of 2 and 3 yields mean latencies approximately double those produced by subitizing. Intriguingly, the mean latencies for quantifying a single dot are much closer (subitizing - 500 msec; counting by ones - 676 msec).

The ERP results represent a tantalising mixture of promise and methodological problems. Data were band pass filtered (3dB attenuation at 0.5 Hz and 45 Hz) and digitized to 12 bit accuracy at a rate of 200 Hz. Initial attempts at process identification and discrimination focussed on analysis of ERP records at the individual trial level. The approach adopted aimed at progressively constructing a process identification template by detecting a salient feature in a single trial ERP record and seeking to find features on subsequent trials onto which it could be superimposed. These efforts were frustrated by the amount of noise present in individual trial records.

An aggregated approach to analysis was adopted. Averaged ERPs were derived for each number of dots in each condition. The averaged potentials in the speeded condition showed a diffuse positivity with a latency of approximately 270 msec. The possibility of deriving an identification profile for subitizing was supported since this positivity was especially prominent in the pre-frontal, central and right parietal regions and increased with the number of dots in the display. These findings are consistent with the hypothesis that subitizing is a primary conscious process, in Edelman's (1989) terms, since he assigns significant roles to pre-frontal, parietal, sensory and motor cortical areas.

The results obtained from the counting condition provide no grounds for optimism that an ERP identification profile for counting by ones will be derived. No consistent features are discernible in the aggregated data. This is scarcely surprising since in the 1-10 numerical range all of the response latencies, except for single dot displays, lie beyond the typical ERP duration of 400 to 800 msec. Indeed, the standard deviations of the latencies obtained with collections of 9 and 10 dots are 880 msec and 1100 msec respectively.

Steady State Evoked Potential Methodology

It is clear that derivation of an identification profile for counting by ones requires a technique that provides more temporally extended data than ERPs. For purposes of comparison it should, ideally, be possible to detect an identifying signature for subitizing using the same technique.

Recent methodological developments may provide what is required. Silberstein (1990) has developed a technique which, for the first time, successfully demonstrates a relationship between the 'steady state evoked potential' and cognitive processes. These evoked potentials are associated with rapidly repetitive stimuli and consist of sinusoidal components at the stimulus frequency or multiples of the stimulus frequency (Regan, 1977, 1989).

The steady state potential has a number of attractive features relevant to the study of cognitive processes. The most significant is the ability to assess the characteristics of the potential in as little as 2 seconds. This makes it an ideal means of investigating time varying phenomena in cognitive processes.

In applying steady state evoked potential methodology to quantification processes the 2 second lower limit on the duration of data gathering necessitates a switch from individual quantifying events to the direct investigation of cognitive modes. In the 1-10 numerical range subitizing and counting by ones when applied to dot collections produce latencies below the 2 second minimum. Accordingly, in current work, subjects are being requested to undertake sequences of quantification comprising extended periods alternately devoted to subitizing and counting by ones. Concurrently, the spatial and temporal features of brain activity patterns are being monitored using the steady state probe paradigm (Silberstein et al., 1990a,b). This will permit an investigation of brain activity patterns associated with shifts in processing mode. Steady state probe topography will be used to explore the dynamics of the correlations between cortical areas during the different quantification processing modes.

Conclusion

Adoption of steady state evoked potential methodology paves the way for investigation of the roles of sub-symbolic and symbolic processing and primary and secondary consciousness in tasks that are more temporally extended than typical, individual occurrences of quantification. As an initial step, two experimental conditions have been selected from those employed by Berry and Broadbent (1988) in their study of implicit/explicit learning and implicit/explicit knowledge. The two conditions have been selected on the basis of their potential to maximize the differences in the respective contributions to task performance of the two modes of processing and types of consciousness.

Introduction of the dimension of learning and acquisition in addition to cognitive performance points the way to a categorisation of modes of cognitive operation that may considerably exceed Norman's predicted minimum of "at least two modes of operation." Future work will apply steady state evoked potential methodology in exploring the distinctiveness of modes defined in terms of different combinations of values derived from dimensions of variation such as implicit/explicit learning, declarative/procedural (compiled) knowledge, primary/secondary conscious operation and processing available/unavailable to introspective report.

References

Berry, D.C. and Broadbent, D.E. (1988). Interactive tasks and the implicit-explicit distinction. *British Journal of Psychology*, 79, 251-272.

Barto, A.G. (1985). Learning by statistical co-operation of self-interested neuron-like computing elements. *Human Neurobiology*, 4, 229-256.

Ciorciari, J., Silberstein, R. B., Simpson, D. G. & Schier, M. A. (1987). The Multichannel Electrode Helmet. *Proceedings of Conference on Engineering and Physical Sciences in Medicine*. Melbourne, 52.

Clark, A. (1989). Connectionism and the multiplicity of mind. *Artificial Intelligence Review*, 3, 49-65.

Edelman, G.M. (1989). *The Remembered Present*. New York: Basic Books.

Gevins, A. S. & Cutillo, B. A. (1986). Signals of Cognition. In Lopes da Silva F H, Storm van Leeuwen W & Remond A (Eds). Clinical Applications of Computer Analysis of EEG and other Neurophysiological Signals. *Handbook of Electroencephalography and Clinical Neurophysiology:* new ser. V2. Amsterdam: Elsevier, 335-384.

Goddard, N. H., Lynne, K. J., Mintz, T. and Bukys, L. (1989). *Rochester Connectionist Simulator*. University of Rochester, Department of Computer Science.

Karmiloff-Smith, A. (1986). From meta-processes to conscious access: Evidence from children's metalinguistic and repair data. *Cognition*, 23, 95-147.

Klahr, D. & Wallace, J. G. (1976). *Cognitive Development: An Information Processing View.* Hillsdale, NJ: Lawrence Erlbaum Assocates

McGillen C D & Aunon J I (1987). Analysis of Event-Related Potentials. In Gevins, A. S. & Remond, A. (Eds). Methods of Analysis of Brain Electrical and Magnetic Signals. *Handbook of Electroencephalography and Clinical Neurophysiology*. Vol. 1, 131-169. Elsevier, Amsterdam.

Norman, D. (1986). Reflections on cognition and parallel distributed processing. In Rumelhart, D., McClelland, J. and the PDP Research Group (Eds.) *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*. Vol II, 110-146. MIT Press, Cambridge, MA, USA.

Reber, A.S., Kassin, S.M., Lewis, S. & Cantor, G. (1980). On the relationship between implicit and explicit modes of learning a complex rule structure. *Journal of Experimental Psychology: Human Learning and Memory,* 6, 492-502.

Regan, D. (1977). Steady-State Evoked Potentials. *Journal of the Optical Society of America,* 11, 1475-1489.

Regan, D. (1989). *Human Brain Electrophysiology: Evoked Potentials and Evoked Magnetic Fields in Science and Medicine.* Elsevier, New York.

Rogers, B.L. (1991). *A New Associative Conditioning Element for Consummatory Classical Conditioning.* Ph.D. Thesis, Swinburne Institute of Technology.

Rumelhart, D. E., Smolensky, P., McClelland, J. L. and Hinton, G. E. (1986). Schemata and Sequential Thought Processes in PDP models. In J. L. McClelland, D. E. Rumelhart and the PDP Research Group (Eds.), *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Vol. 2, 7 - 58. MIT Press. Cambridge, MA, USA.

Silberstein, R.B., Schier, M.A., Pipingas, A., Ciorciari, J. and Wood, S. (1990a). Dynamics of the steady state visually evoked potential topography in a visual vigilance task. *Brain Topography.* 3: 272-273.

Silberstein, R. B., Schier, M. A., Pipingas, A., Ciorciari, J., Wood, S. & Simpson, D. G. (1990b). Steady state visually evoked potential topography associated with a visual vigilance task. *Brain Topography,* 337-347.

Starkey, P. and Cooper, R. G. Jnr. (1980). Numerosity Perception in Human Infants. *Science* 210, 1033.

Wallace, J.G. & Bluff, K. (1989) *Knowledge Acquisition and Cognitive Abilities: an Information Processing Approach.* Final Report to ARC.

How Students Construct a Shared Understanding of Collisions in Newtonian Mechanics

Denise Whitelock, Josie Taylor, Tim O'Shea, and Eileen Scanlon
Institute of Educational Technology
Open University, Walton Hall, Milton Keynes, MK7 6AA, England

Paul Clark
Physics Faculty
Open University, Walton Hall, Milton Keynes, MK7 6AA, England

Claire O'Malley
Department of Psychology
University of Nottingham, University Park, Nottingham, NG7 2RD, England

Abstract: Two families of causal models appear to develop to help pupils explain the motion of objects after a collision. These are a *linear causal* model and a *resistance/reciprocal causal* model. These models however, break down when the students are confronted with the computer simulation, if conservation of energy is not understood. The computer simulation allows pupils time to run experiments which they think will confirm their faulty models but in fact facilitates a shared understanding of energy conservation. Our hypothesis, arrived at as the result of the pilot study, is that pupils working collaboratively with a computer simulation can progress towards a causal model which is more like that of the physicists i.e., a "resistance type" model.

Introduction

The Computer Supported Collaborative Learning in Physics study is a joint project between the Institute of Educational Technology and the Physics faculty within the Open University, and the Psychology Department at the University of Nottingham. Our general aim is to develop computer software which supports collaborative learning between pairs of students solving physics problems. We could not however at this early stage in the project tackle collaborative learning, software development and physics learning in the same depth simultaneously. Therefore, a pilot project, originally undertaken in two schools, was designed to collect some basic information about the level of physics knowledge of dynamics with respect to elastic collisions.

The preliminary findings (Whitelock, Taylor et al., 1991), indicated that most of the pupils, from one particular school, were aware of the physics formulae and could solve two thirds of the prediction problems. We found this result surprising because previous studies, for example, Lawson and McDermott (1987) and Villani and Pacca, have shown it is an area which is not well understood. Stainton-Ellis, O'Shea and Scanlon (forthcoming) report a great diversity of responses, from both adults and school children alike, who seem to be applying models that were internally inconsistent to the problem, as well as exhibiting some rather strange intuitions about the physical world.

We therefore extended the pilot study to include a further seventeen 'A' level Physics students from a third comprehensive school and were interested in examining the combined data, with respect to the following questions:

1. How well is the physics understood?
2. Do students apply commonsense models of reasoning to the problem?
3. What happens when students with different models use the computer simulation?
4. How is a shared understanding reached?

Method

The investigation was divided into four phases.

Phase 1. Completion of Questionnaire

This included a prediction task where subjects were asked to predict the subsequent motion of two ice pucks after collision for the following three cases:

Case 1: Puck A is small and light; Puck B is larger and heavier.
Case 2: Puck A is large and heavy; Puck B is smaller and lighter.
Case 3: Puck A and Puck B are identical.
N.B. Puck A always hits a stationary Puck B.

The subjects were questioned about their understanding of kinetic energy and momentum. Most of these later questions were taken from, or adapted from, the APU Science in schools: Age 15 report, (Welford et al., 1983).

Phase 2: The Interview

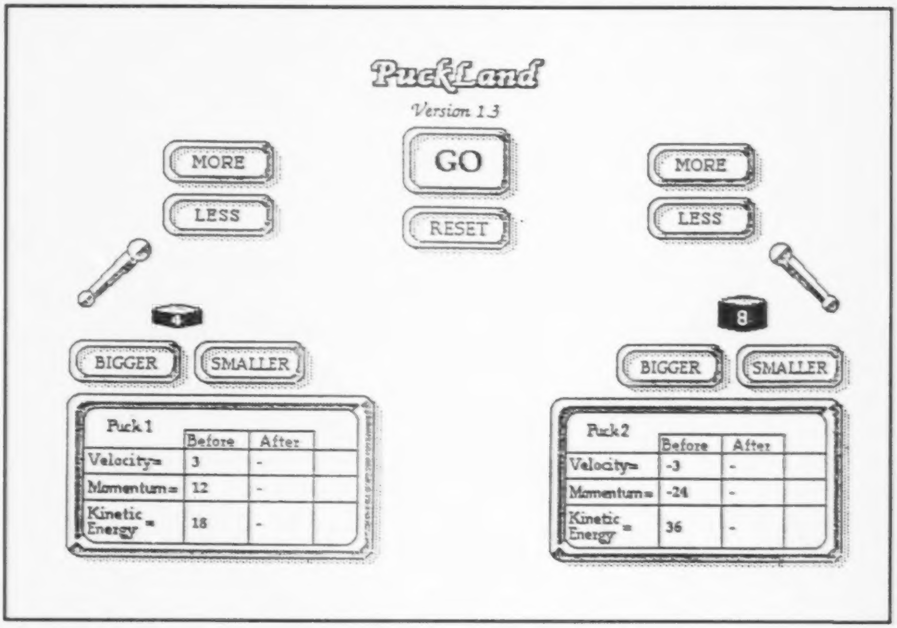
In order to understand how the subjects make sense of the movement and impact of objects, prior to working in pairs with the computer, a clinical interview technique was adopted. The interview needed to be structured in such a way that subjects felt it was both reasonable and possible to explain what is often most obvious and would normally require little, if any, explanation. Therefore a number of individual interviews focussed around a series of events all involving collisions as illustrated in children's comics, a technique previously used by Bliss, Ogborn and Whitelock (1989).

Phase 3: The Co-operative Phase—Subjects Working in Pairs

The subjects worked in pairs and had access to their paper and pencil task predictions concerning the subsequent motion of the pucks after collision. They were asked to discuss their predictions and to try and sort out any differences between them. Then each pair used the computer simulation where they had time to explore any situations which interested them.

The Puckland simulation, developed by R. Sellman, used a direct manipulation approach, (Smith et al., 1989), which allowed students to investigate a series of collisions between two ice pucks. It was written in Hypercard for use with the Apple Macintosh and consisted of a pair of pinball-style flippers on either side of the screen with which subjects could flick pucks (see Figure 1). Subjects could vary both the mass and the initial velocities of the pucks whose apparent screen velocities after collision were calculated from the appropriate physics formalisms.

Finally the students were asked to solve three different problems, which were posed in order of difficulty. The pupils were not required to predict the *result* of a collision but to predict the *original conditions* for a given outcome. The problems were as follows:



Puck 1	Before	After	
Velocity=	3	-	
Momentum=	12	-	
Kinetic Energy =	18	-	

Puck 2	Before	After	
Velocity=	-3	-	
Momentum=	-24	-	
Kinetic Energy =	36	-	

Figure 1. Screen Dump of the Graphic Interface to the Puckland Program

1. What initial conditions are needed to send the pucks travelling away from each other at the *same speed*?
2. What initial conditions are needed to make one puck *stop* after impact?
3. What initial conditions are needed to make pucks of *unequal* masses move away, after impact, at the *same speed*?

Phase 4: Teachback Phase

The purpose of this phase of the computer simulation was to ascertain what the pupils had learnt and so they were asked to describe what important factors should be taken into account in order to perform the prediction task correctly.

Sample

The total number of students studying 'A' level physics, in the three schools was 37. Although all these subjects answered the prediction task and questionnaire only half the sample, i.e., seven pairs volunteered to interact with the Puckland program.

Results

The Prediction Task

Only five of the subjects successfully predicted the correct motion of the ice-pucks for all three cases tested. Three of the pupils answered all three cases incorrectly. However, 72% of the pupils tested were able to answer CASE A correctly. With only 45% correctly answering CASE B and 48% predicting CASE C accurately as shown in Table 1 below. This final result suggests that less than half the pupils could be using the conservation of energy principle in their reasoning about motion. These results differ from earlier paper (Whitelock et al., 1991) where only twenty subjects were tested. It appears that the students from one of the original schools tested were more competent than the other two schools and this meant that pupils were able to answer CASE A and CASE B correctly and none of them got all three predictions wrong.

Physics Knowledge Elicited from the Questionnaire

Subjects were asked to identify the factors which they considered to be important in predicting the motion of the ice-pucks after impact. Ninety percent of the pupils indicated that mass was important. Seventy percent considered velocity to be a saliency factor. These variables would be needed to calculate both momentum and kinetic energy if one was going to use a "physics framework" to make the predictions.

However when asked the formula for calculating velocity only 50% of the students realised it was distance divided by time. Forty-six percent could recite the formulae for kinetic energy and momentum although most of the subjects realised momentum had "something to do with" mass and velocity. When asked what is meant by kinetic energy, half the students thought it was some sort of force which could be stored, while only a quarter of them associated it with movement.

Physics Knowledge Elicited from the Comic Strip Interviews

Although only a third of the sample were interviewed, these results do give us some idea of the types of physics knowledge pupils spontaneously use to explain collisions, when the questions are framed in a non-school setting. The most frequently used terms were speed, force, energy and momentum. The interviews revealed that the physics terminology was used very loosely with students offering their own "commonsense" definitions for the above terms (Whitelock, Taylor, O'Shea, Scanlon, Sellman, Clark and O'Malley, 1991). It appears that the rigorous definition attached to these words by the physicist is not always found in an "everyday understanding," a finding also reported by Whitelock (1990).

Knowledge Elicited from Explanations of Prediction Responses

The transcripts were analysed to see if there were any similar reasoning patterns between the students to explain the motion of the ice-pucks after impact. There appear to be two different models used by the students namely a *linear causal model* and a *resistance/reciprocal causal model*. With the linear causal model, the prime cause and predictor of motion is the mass of the incident puck. Whereas the resistance model does not just utilize a primary causal agent to explain motion but also takes into account the presence of the

Table 1
Results of Prediction Task

N = 37

	CASE A		CASE B		CASE C	
TOTALS	Fs/Ff*	27	Fs/Ff*	17	St/Fss*	15
	Fs/Fss	5	Bf/Fs	4	Bss/Fss	1
	Bs/Fs	1	Bf/Fs	2	Bss/Fss	4
	Bs/Ff	2	Bf/Sta	1	Bs/Fs	1
	Sta/Ff	1	Bs/Sta	2	Bss/Ff	1
	Fs/Bf	1	Bs/Fss	1	Fs/Ff	9
			Bs/Fs	2	Fs/Fss	2
			Sta/Fs	2	Fs/Fs	1
			Sta/Ff	2	Fss/Fss	2
			Fs/Fs	1	Fs/Bf	1
			Fs/Ff	2		
			Fs/Bf	1		

CASE A = LARGE MASS HITS SMALL MASS
CASE B = SMALL MASS HITS LARGE MASS
CASE C = EQUAL MASSES HIT EACH OTHER

St = Stationary F = Forward s = Slower
B = Backward ss = Same speed f = Faster
*CORRECT

target puck. The subsequent motion is then thought about in terms of the resistance which can be offered by this second puck.

However both these models "break down" or do not produce the correct predictions, when conservation of energy is not understood. This means that although both models predict a transfer of energy from a moving to a stationary object they cannot say how much energy/force is transferred, if the above principle is not understood. This accounts for the incorrect predictions when pucks of equal masses collide.

A student who used a *linear causal model* and did not understand the conservation principles made the following predictions.

CASE A Fs/Ff*
CASE B Bs/Sta
CASE C Fs/Ff

His argument was as follows:

1. When two objects collide energy is transferred/exchanged.
2. The size (mass) of the objects determined how much energy can be given or taken during this exchange.
3. A larger mass has a large amount of energy and can donate a large amount of energy.
4. A large mass will only lose all its energy to a bigger mass than itself.
5. A large mass loses only some of its energy to a smaller mass than itself.
6. However a small mass does not lose all its energy to a larger mass and remain stationary. It rebounds backwards because a small mass cannot move the big mass at all. The motion does not just stop.
7. When two equal masses meet not all energy is transferred because a bigger mass is needed to absorb the energy from any moving mass.
8. Objects move in the direction of their initial push/force unless something blocks their motion e.g., a ball rebounds from a wall.

A student who used a *resistance causal model* made two correct predictions but again did not understand the conservation principles. His predictions were:

CASE A Fs/Ff*
CASE B Bs/Fs*
CASE C Bs/Bs.

He explained these predictions in the following way:

1. Energy is transferred during a collision, this is kinetic energy.
2. Energy is absorbed by the stationary puck during collision.
3. When a large mass hits a small stationary mass it has enough momentum to carry itself on and to move the little mass.
4. Small masses cannot overcome the larger mass inertia and bounce back.

These two different models exhibit a change in causal thinking. They do not allow correct predictions if conservation of energy is not understood but use of the resistance model is a move towards a physicist's way of reasoning about the problem.

What Happens When Students with Different Models and then Similar Models of Reasoning About Collisions Use the Computer Simulation

Three case studies have been selected to illustrate what happens when students with different opinions, work with the computer simulation. Case studies one and two, illustrate a difference not only in *control* of the dialogue but also in the *types of experiments* performed with the simulation In case study one, for example, the students do not adopt a systematic approach to control of variables. Their whole set of experiments is driven by one of the partners who does not understand as well as the other. However the "more able" partner allowed the "less able" one time to run his own experiments when he could not understand the explanations the "more able" one gave. In these two case studies, conservation of energy was not initially appreciated by both partners but was understood at the end of the experiments.

In Case Study three both partners made initial correct predictions about the motion of the ice pucks and when they paid immediate attention to the qualitative data given by the experiment. They subsequently rediscovered the formulas for kinetic energy and momentum. There was more equal power sharing between these partners.

In general, for all three couples a more definite exchange of their "commonsense understanding" of motion took place during the problem solving phase with the computer and for the pupils in Case Studies 1 and 3 both partners moved towards thinking about the problem, in terms of a resistance, rather than a linear causal model.

What Happens When Two Students Who Do Not Understand the Conservation of Energy Principle Use the Computer Simulation

CASE STUDY ONE

These students made the same accurate predictions for CASES A and B but not for CASE C. They thought pucks should move backwards at the same speed for CASE C instead of one remaining stationary. After trying out their first two predictions with the simulation and finding they were right, they were genuinely surprised that they had predicted CASE C incorrectly. They repeated the experiment giving the initial puck more velocity believing this would confirm their initial prediction. They were both surprised and suggested there was a fault in the program.

Iv. I would have thought it would have carried on moving.
B. I would have thought that as well.
Iv. It is the way it has been programmed because it disappeared.

While trying to find out why they were both wrong, one of them noticed that conservation of energy and momentum was recorded with every impact.

B. If you look at the results, the results say that before and after the velocity, the momentum, the kinetic energy were the same. If you add the results up with the pucks.

The partner who noticed this was able to predict when equal masses and equal velocities hit they would move backwards but the other partner predicted they should both stop dead. His partner's explanation however was not helpful to him and he had to try the experiment again before he realised his partner was correct.

Iv. No. they can't can they. Oh hang on, if ...? yeah they can't the only way it can really happen is if one is already stationary. You can't get them (both) to stop because the momentum is conserved.

After the pair tried out the cases given in the pre test they started to look for a relationship between initial mass and final momentum. They tried a number of experiments. Varying initial mass they found:

Iv. So far the pattern is when it is three times the size the momentum is halved, when the puck is four times the size of the one it hits the momentum is four thirds, when it is five times the size it is three halves.

The pattern breaks down with further explanation and the partner who dominated this part of the session remarks:

Iv. We are getting different values now.

B. Yeah.

Iv. Gives us one and two sixths, so obviously what we worked out doesn't seem to work, oh well.

The other partner takes control again suggesting they investigate the change in kinetic energy between the pucks. They experiment and conclude there is a relationship but they cannot find it.

Iv. Yeah it tried, It tried, I know it is connected to kinetic energy and momentum but I can't remember what it is.

During the course of the investigating phase one partner dominated the proceedings. In fact the experiments chosen were driven by his hypotheses. The second boy realised his explanations would not convince his dominant partner and although he always tried to explain he realised that his partner would have to experiments for himself before he became convinced of his reasoning. However in the problem solving phase there was more equal exchange of dialogue.

When asked what they had learnt from the simulation they replied:

Iv. We almost had one rule but.

B. But all, all velocity (...?) kinetic energy all conserved, there is none lost from the total at the beginning and it equals the total after the.

What Happens When Two Students Who Disagree About CASE C Use the Computer Simulation?

CASE STUDY TWO

These Students made the same predictions for CASE A and CASE B but disagreed about CASE C. In fact one had some understanding about the conservation of energy while the other did not.

D. Right, the two small ones, I had the small one moving forwards slower and that the second one would move forwards faster than the first one.

N. That is different, I have got my first one would hit and stop and the second one would move forward faster ... I thought they were the same mass then all its energy could be transferred to the second one.

D. Yeah.

N. Throw all of it into it so the first one would lose all its energy and stop.

D. Yeah, well I think we will have to see about that one.

They found out during the course of the experiment that one of them was right but they did not explain the correct result any further.

Although, like the previous pair, they were looking for some underlying relationship between initial mass and velocity to predict the subsequent motion of the pucks, they experimented more systematically with the controllable valuables. They started by noting the effect of change of mass.

D. Right, so let see. If we start off with say, puck of a certain size, one this side, yeah?
N. Right.
D. And then we try all different sizes of puck on the one that is moving and see where the point is that one go faster where it stops and where it goes slower yeah.

They then varied the velocity of the moving puck systematically, leaving the mass constant.

They noticed that conservation of energy and momentum occurred during the course of their experiments.

D. The combination of the velocities added together afterwards makes the same as the initial velocities added together afterwards. The momentum is four, the momentum stays the same, and the kinetic energy stays the same as well. Maybe we need to work on the proportions.

They looked for a simple proportional relationship, like the couple before them.

D. Yeah. All we need really is to find out what it has to do in relation to mass and velocity because momentum and kinetic energy are just changing because the mass and velocity are changing.........
So if we do it with X mass and then try it with half X, 2 X, 3 X, whatever yeah.
N. And see, proportion.
D. Proportion of the velocity and mass, and so yeah. We will start of with X mass. So if we start off with something in the middle like four.

This pair adopted a more systematic investigation but reached similar conclusions to the previous pair in that they couldn't find the connection between momentum and velocity and kinetic energy. However they quoted some rules. This practical work is perhaps preparing them to accept a quadratic relationship?

N. The same mass with one double the velocity, we worked out the same mass the velocity was exchanged each time wasn't it?
D. Yeah, so they repelled each other but the other had the other velocity.
N. Um we found that the same velocity but three times the mass we got a stop, the last one we just did we had a double the mass going half the speed we got I don't know, we had double the mass going double and half the mass going half the velocity, we got a stop there as well.

What Happens When Two Students with Similar Models Use the Computer Simulations? (Models Predict Correct Motion of Pucks)

CASE STUDY THREE

Both couples agreed about their predictions and one might assume that they understand the conservation of energy principle from their responses but in fact when they discuss their results they admitted that they relied on their experience of playing pool to make the predictions for CASE C.

A. *Well the first puck comes along hits the second puck, the first puck stays where the second puck was and the second puck goes off with the same speed as the first puck,* ***it worked on a pool table.***

They ran test CASES A, B and C first and found their predictions were correct. They both paid attention to the qualitative data given during the experiments.

A. *So after the collision the first mass carried on moving forward with a velocity of 1.5, so it has halved its velocity in that.*
S. *Um. And this one is going at 4.5*
A. *Yeah.*
S. *So that is the original velocity, minus 1.5 isn't it. No. the original velocity of puck one plus 1.5 that it has lost.*
A. *Yeah.*
S. *Because look that velocity goes 4.5. Now its momentum is the same, I think it stays, yeah that has gone up it has gone to nine, it has halved that. Yeah.*
A. *As the speed has halved the momentum has halved.*

There was more power sharing during the exploratory phase with the computer and they agreed amongst each other which experiments to explore and they also experimented with the limiting conditions.

S. *Yeah, right, what are we going to try now, say, how about say really big, with not much speed travelling again say pretty small with quite a bit of speed?*

This pair worked out momentum equals mass x velocity and rediscovered the formula for kinetic energy.

When problem solving their original models are more evident than in the exploratory phase.

S. *So it would be impossible to get them to both travel at the same speed in the same direction.*
A. *Yeah.*
S. *Because the ball ...(they keep referring to pool games).*
A. *We haven't got it any where down here.*
S. *No but as far as I can see, the first puck is going to have to transfer some of its energy, so therefore it is going to be slowed down, so it won't be going as fast as it originally was to get the first one, the second one moving.*
A. *Yeah.*
S. *There's a third.*
A. *Got it.*
S. *So do you want to try, transfers ...*
A. *Transfers some of its, in the last one it hit the puck two didn't it?*
S. *Yeah.*
A. *And the speed it was then travelling at, it was added on to there. Um, so if you have got a velocity of two and a velocity of four, if you have puck two as a bigger mass than puck one.*

S. Yeah. Try that.
A. Put puck one as say two and that as four, and that moves at a velocity of four. Try that.
S. Yeah.
A. Something ... How would we stop it actually rebounding, if we could give it more speed, the puck would just come back quicker.
S. Um, we need more momentum, more mass.

How Was the Shared Understanding Achieved?

Roschelle and Behrend (forthcoming) propose a framework for analysing collaboration which includes turntaking with the computer controls and in conversation. There is some evidence to suggest that conversational turntaking does include the use of ongoing narratives, questions and repairs to maintain shared intelligibility and continuation of the task in hand. These are however more prevalent in the problem solving phase. There is another strategy used during the exploratory phase and that is the "more able" partner allowed the "less able" one time to run his own experiments if he could not understand the explanation given, i.e., they allowed one another space and time to "catch up" on their understanding--so as they could then move forward together.

When the subjects participated in the problem solving phase of the computer interaction their pattern of collaboration changed. This alteration of behaviour was particularly marked when the subjects were asked to predict what conditions would be needed to produce pucks of unequal masses to travel at the same speed after impact. With respect to this problem a number of factors changed.

1. Both subjects did not think or assume they knew the answer and both genuinely wished to solve the problem.
2. The person controlling the computer was not totally sure what to do and so was more willing to seek advice from his partner.
3. A fuller discussion took place after the simulations were run because both partners genuinely found the computer's response surprising.

This phenomenon however merits further investigation as the observations reported here must be cautiously regarded because of the small numbers involved.

Conclusions

One thing the computer simulation did reveal to the less competent students, was the principle of conservation of energy, which the students noticed from the qualitative data given after every collision and from their incorrect predictions about the motion of two ice-pucks of equal masses. Our results suggest that two families of causal models can develop to help pupils explain motion after a collision and our hypothesis, arrived at as the result of this pilot study, is that pupils working collaboratively with a computer simulation can progress towards a causal model, which is more like that of the physicists i.e., a "resistance type model."

We found that when the subjects first used the Puckland program, they were unwilling to discuss the computer's responses with respect to their own ideas, as they were more intent in trying to understand the scientific model. However, when they were given specific problems to solve a more definite exchange of their "commonsense understanding" of motion took place.

The advantage of using a computer simulation in this study was that it gave the students time to run their own experiments which were based on their own understanding of the pucks problem. Simulations can provide a source of reliable information if the students believe they represent the scientific model as usually demonstrated by suitable physical apparatus in the science laboratory. Simulations also provide a potential mechanism for exploring socio-cognitive conflict, an area we are currently investigating as an extension to this pilot study.

Acknowledgements: The research reported in this paper was funded by ESRC grant R000231660 to Tim O'Shea, Eileen Scanlon Paul Clark and Claire O'Malley. The authors wish to thank the ESRC for their support and Royston Sellman for the preparation of the software.

References

Amigues Rene, (1990) *Semiotic Tools, Social Interaction and Collaborative Talk in Instructional Settings: A problem of reference?* International Conference "Social Interaction and Knowledge Acquisition in educative settings." Rome, 4.

Bliss, J., Ogborn, J. and Whitelock, D. (1989) Secondary Pupils' Commonsense theories of motion. *International Journal of Science Education.* Vol. 11. No. 3.

Lawson R., and McDermott L. (1986) *Student understanding of the work-energy and impulse-momentum theorems.* 1987 American Association of Physics Teachers.

Roschelle J., and Behrend S. (forthcoming) *The Construction of Shared Knowledge in Collaborative Problem Solving.* C. O'Malley (Ed) Computer Supported Collaborative Learning.

Smith R.B., O'Shea T., O'Malley C., Scanlon E., and Taylor J. (1989). *Preliminary Experiments with the Distributed, Multi-Media Problem Solving Environment.* CITE report No. 86.

Stainton-Ellis, K., O'Shea, T., and Scanlon, E. (forthcoming). *"Billiard Balls, Golf Clubs and Magnets: Models of Collisions in Newtonian Mechanics"* CITE report in preparation.

Taylor, J., O'Shea, T., Scanlon, E., Sellman, R., Clark, P., O'Malley, C. (1990) *Preliminary Findings in Computer Supported Collaborative Learning in Physics.* CITE report No. 118.

Welford, G., Bell, J., Davey, A., Gamble, R., and Gott, R. (1983) *APU Science in Schools: Age 15.* Research Report No. 4.

Whitelock, D. M. (1990) *Commonsense understandings of causes of motion.* Unpublished Ph.D. Institute of Education, University of London.

Whitelock D., Taylor J., O'Shea T., Scanlon E., Sellman R., Clark P and O'Malley C. (1991) *Collaboratively Debugging Shared Models for Energy Conservation using a Computer Simulation.* PEG91, Knowledge Based Environments for Teaching and Learning, Genova.

Villani, A., Pacca, J.LA. (1990) Spontaneous Reasoning of Graduate Students. *International Journal of Science Education,* Vol 12, No. 5, 589-600.

Principles and Practice of Knowledge Representation in EUROHELP

Radboud Winkels, Joost Breuker and Nienke den Haan
University of Amsterdam
Dept. of Law & Computer Science
Kloveniersburgwal 72, 1012 CZ Amsterdam, The Netherlands

Abstract: This paper discusses the development of a knowledge representation formalism for modelling interactive computer applications for an Intelligent Help System (IHS) Shell called EUROHELP. Ideally a developer of a future Help System should only have to fill the shell with the objects and system procedures of the particular application to construct the IHS. This approach has lead to a number of requirements for the structure of this application model (AM): It should be generic–covering all kinds of applications–and multi-functional, i.e., it should serve the various active modules of the EUROHELP shell, like planners, a diagnoser, a coach, etc.

In principle an application can be viewed as a type of device, with a structure and (functional) behaviour. Therefore, developing an ontology which covers the working of all these types of applications should provide this multi-functionality and scope. We present such an ontology, based upon notions of sets and ordered sets, and their basic operations, leading to the definition of canonical actions. This principled and deep reasoning approach could be a first step towards a Qualitative Computer Science.

However, it alone is not a practical solution. A more operational view of the functions of the target application is provided by a hierarchical task layer in which the mode concept plays an important role. The resulting combined formalism for Application Modelling fulfils the requirements mentioned.

Introduction

EUROHELP was an Esprit project (1984-1990) which produced a shell and methodology for developing Intelligent Help Systems (IHS) for interactive computer programs. This shell is conceived in such a way that, ideally, the development of a help system for a particular application consists of filling the domain representation (Application Model, AM) with the objects and system procedures which constitute the particular system. A EUROHELP IHS has both a passive side–i.e., the user can ask questions–and an active side, where the IHS is in fact a coaching ITS which looks over the shoulder of the user when he is using an application, e.g., a text editor, mail system, etc. (Breuker et al., 1987; Breuker, 1990). In the first case, the Question Interpreter will try to define the users need, in the second case the Performance Interpreter will have to compare its ideas on the intentions of the user with the actual performance. This is achieved by a Plan Recognizer and a Planner. Both Question and Performance Interpreter can consult a User Model to help them identify possible problems or needs for information. Once such a need has been identified, the Diagnoser will have to 'explain' it in terms of a lack of knowledge or a misconception on the part of the user. If it does find such an explanation, it is sent to the Coach, who will plan a sequence of communication actions to deal with it (Winkels et al., 1988).

All these active modules will need to access a knowledge base describing the particular application (the Application Model). It is the structure of the Application Model we will discuss in this paper.

Principles of Application Modelling

In selecting and designing the formalisms for Application Modelling in EUROHELP two important requirements were identified from the start:

Generic structures: The formalisms should be both generic and specific for Information Processing Systems (IPS, i.e., applications like Vi, Mail, etc.). They should be sufficiently general to cover this large variety, and more specific than a straightforward knowledge representation language, because it is assumed that the world of computer applications is more specific than any 'real world'.

Multi-functional formalisms: Because a help system has to provide a multitude of functions it would be most convenient if a single formalism could be employed to represent all objects and system procedures of an IPS. The various active modules of EUROHELP, like the diagnoser, plan recognizer, coach, could be viewed as containing specific interpreters that generate new views or structures from this AM. The alternative is to use specific formalisms for each (class of) functions, e.g., by having each module have its specific knowledge base. This latter solution is rather the current wisdom in designing complex systems, but not very practical in our case. A single AM knowledge base which contains all required domain knowledge by a uniform formalism has the important advantage that it is coherent, easier to check with respect to consistency, more parsimonious and maintainable, e.g., the meaning of concepts is not distributed all over the various modules of a system. It is also easier for a help system developer to fill and edit one AM and he can be further supported by providing him with initial, generic models of IPSs.

The requirements of multi-functionality and generality lead to the same principle: the specification of the structuring elements that constitute the world of IPSs. From the generality point of view it is hypothesized that this world is a rather *'narrow'* one, as is the case with most 'artefactual' systems relative to 'natural' ones. From the multi-functional point of view the AM should be sufficiently *'deep'* as to allow interpretations that drive planning, diagnosing, etc.

Operational versus Support Knowledge

The notion that 'deep' models support a larger variety of reasoning strategies has been defended by many researchers (e.g., Clancey, 1983; Davis, 1983; Wielinga & Breuker, 1986;

Steels, 1987), but there is little evidence how wide this range is. There is on the other side Chandrasekaran's (1987) 'interaction hypothesis', which states that this scope is probably limited to a single reasoning regime, i.e., the structure of knowledge is always geared to its particular use. He does not distinguish between deep or compiled-out knowledge. For the latter type his hypothesis is almost by definition true. Aside from the scope and 'depth' of knowledge, there is little doubt that declarative formalisms, which separate the specification of the knowledge from its use, are to be preferred to procedural ones; not only because their acquisition and editing is more simple, but in particular because the former supports a wider inference capability (Genesereth & Nilsson, 1987).

Although different points of view are involved, the distinction between declarative and procedural knowledge representation correlates strongly with two types of knowledge: Clancey's (1983) distinction between operational vs. support knowledge.[1] Support knowledge consists of the models of a particular domain which allow one to infer *what* and *why* it is the case in a domain. The operational knowledge can be conceived of as a particular restructuring or compiling-out of this knowledge by experience in particular types of tasks. These experience based associations can be conceived of as instructions on *how* to accomplish these tasks.

In principle support knowledge is sufficient to describe a system and derive all potential behaviour: it constitutes the deep, 'qualitative' model of a domain. However, for IPSs this may be problematic:

- Potentially far too much behaviour can be derived, i.e., more than the original intended functionality of the system.
- The same behaviour (effect of a system procedure) may have multiple intentions. One may delete a file to obtain space for some other file, or to prevent confusion with older versions, or to annoy another user, etc. These potential external effects–i.e., intentions of a user beyond direct manipulation of a system–cannot be captured by a model of the system.

Practically this means that the AM should contain both operational and support knowledge. Operational knowledge consists of task structures of users and is of practical importance because it constrains the size of the interesting 'behavioural' space of the system to a limited number of functions. Another reason to have prespecified typical tasks wired in the AM is that a task (e.g., sequence of behaviours) may have functions in the world external to the IPS. The name of a task structure is an interface to this external world.

Representing Support Knowledge

The functioning of the IPS has been modelled by representing both the objects available within an IPS and the way in which these objects can be changed by actions explicitly (see also: Breuker, 1988; Duursma, 1988).

Objects

The term *object* denotes all pieces of information a user handles during the performance of tasks within an IPS, such as rules, messages, words, files, etc. The description of the state of an application consists of several specific instantiations of objects. Objects are

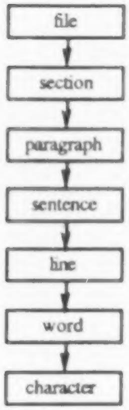


Figure 1. Object Hierarchy

characterized by their attributes and their relations to other objects. Classes of attributes are: *identifiers* (since objects may be referred to, e.g., by their name), *properties* (describe the features of an object), and *options* (define which actions are allowed or prohibited on the object).

There are two kinds of relations between objects. The first is the is-a, kind-of, etc. relation that is used for typing objects (classes and sub-classes). The second kind of relations are *set* relations. Unordered set relations are the has-part relations: contains (for special information container objects), and consist-of (for other objects). The has-part relation defines the ordering of objects in a hierarchy, the so-called *object-hierarchy*. This hierarchy (Figure 1) reflects the major information-structure within most IPSs. The ordered set relations can take various dimensions and can be metaphorically characterised as spatial relations (e.g., before, after, above).

Actions

The actions are defined on the basis of the object definitions. Actions may either change the contents of an object, or its relation to other objects. The action types are also generic: they are defined on certain types of objects, but when used on the state of an application they are translated to changes of object-instantiations. Since all object-instantiations together constitute the state an application is in, an action can be looked upon as an alteration of the state description. An action consists of the description of its antecedent state, and its consequent state. The action itself is synonymous to its *effect*, because an action uniquely describes the alteration of objects or their relations.

We defined a set of primitive actions (Table 1) that was not meant to be exhaustive, bu which was sufficient to describe all situations:

Primitive actions can be combined into composed actions, which in turn are part of system procedures. System procedures (SPs) represent the system facilities available in an application, and are the smallest units of functionality that can be invoked independently by the user. System procedures which are not directly accessible to the user, but can only be invoked indirectly, are called procedures. An SP contains a number of composed actions and some additional procedures. This means that the overall functionality of an SP is described by its members. However, a system procedure does not contain any control statements. The control over the system procedures stems from higher regions.

Representing Operational Knowledge

The representation of the operational knowledge is clustered around a set of tasks. These tasks represent user intentions: tasks users intend to perform by the use of the application. How these tasks can actually be performed is represented by plans containing sub-tasks in a multi level model. Basically the representation formalism is an AND/OR tree with added control and parameter passing constructions.

The EUROHELP model contains tasks at three levels that resemble those of Moran's Command Language Grammar (Moran, 1981):

Table 1
Primitive Actions

<table>
<tr><th colspan="2">On unordered sets</th><th colspan="2">On ordered sets</th></tr>
<tr><td>create(Object)</td><td></td><td>add:</td><td>create(Object)</td></tr>
<tr><td>delete(Object)</td><td></td><td></td><td>Set = S1|S2</td></tr>
<tr><td>add:</td><td>Set := Set ∪ Object</td><td></td><td>Set := S1|Object|S2</td></tr>
<tr><td>remove:</td><td>Set := Set \ Object</td><td>remove:</td><td>Set = S1|Object|S2</td></tr>
<tr><td>move:</td><td>Set1 := Set1 \ Object</td><td></td><td>Set := S1|S2</td></tr>
<tr><td></td><td>Set2 := Set2 ∪ Object</td><td></td><td>delete(Object)</td></tr>
<tr><td>change:</td><td>change-attribute(Object.attribute,-)</td><td>move:</td><td>remove(Object,(S1|Object|S2))</td></tr>
<tr><td>replace:</td><td>remove(Object1)</td><td></td><td>add(Object,(S1|S2))</td></tr>
<tr><td></td><td>add(Object2)</td><td>copy:</td><td>add(Object,(S1|Object|S2))</td></tr>
<tr><td>instantiate:</td><td>add(Object,Selection)</td><td></td><td></td></tr>
<tr><td>delete instantiation:</td><td>remove(Object,Selection)</td><td></td><td></td></tr>
<tr><td>show:</td><td>copy(Object,Screen)</td><td></td><td></td></tr>
<tr><td>hide:</td><td>remove(Object,Screen)</td><td></td><td></td></tr>
<tr><td>open:</td><td>change-attribute(Object.options,open)</td><td></td><td></td></tr>
<tr><td>close:</td><td>change-attribute(Object.options,closed)</td><td></td><td></td></tr>
</table>

1. **User task level**: Here stereotypical user tasks for a class of applications (e.g., text editors) are mapped onto application specific tasks (e.g., Vi). At the bottom of this hierarchy those tasks are represented that are directly supported by the application (the so called mode-tasks, because these are the tasks that can be carried out within one specific mode).
2. **System procedure task level**: Here the mode-tasks are mapped onto a sequence of system procedures tasks, which invoke system procedures.
3. **Interaction task level**: Here the system procedures are mapped onto a sequence of interaction tasks, which represent the physical actions the user can perform on the system. This level describes pressing keys and clicking mouse buttons.

Conclusions

Towards a Qualitative Computer Science

The approach to domain modelling discussed focuses on the representation of the support knowledge in terms of objects and actions in the world of IPS. As in modelling physical systems it was assumed that the definition of actions (system procedures) and objects (and their specific attributes and the way they were structured) of an IPS could be described completely at the "knowledge level" (Newell, 1982). Animation, i.e., interpretation of the structural description of these objects and actions of an IPS would generate a full *behaviourial* description of the IPS. The basic interpreter for this model of models of IPS should consist of definitions of the *primitive actions*. These primitive actions have an *epistemological* status, i.e., they should have semantics, at least in terms of a machine interpreter (Brachman, 1979). The set of primitive actions presented here is based upon general notions: i.e., set theory (see also Sathi et al., 1985 for a KR system based upon set theory). It has the advantage that its foundations are very clear. Its set theoretical roots provide well established semantics. However, we have only limited experience in applying these.[2]

Well established semantics of actions for a set of primitive actions allow for standard interpretations of the effects of system procedures. Primitives are also supporting elements for consistent knowledge acquisition, i.e., in editing the effects of system procedures. An important side effect of having primitives in the EUROHELP KR is that they can map easily on a standard vocabulary or dictionary of action terms (verbs); in this respect the primitive actions have a similar status as those in Conceptual Dependency theory (Schank, 1975).

A Functional View

Generating all possible behaviour from a structural description easily leads to a combinatorial explosion of possible states of an IPS. This not only calls for the right levels of abstraction, but also for a view on what is *functional* or intended behaviour. In general that is the kind of behaviour the designer of the IPS was aiming at. As discussed above, this intended behaviour should be made explicit as operational knowledge in terms of task structures. These task structures include both the level at which the user plans his activities, and the interaction level, which specifies the actions the user has to perform to make the application do what he wants. This limits the space of the potential intentions of users, and avoids generating task structures for all potential (goal) states the application can get into, given its current state. A secondary, but as important benefit is that the ready made plans in the AM allow for *plan selection*, instead of constructing plans from scratch (the system procedures). This makes the planning process far more efficient.

The combined formalism for representing operational and support knowledge also supports other functionality of Help Systems, like automatic curriculum generation, diagnosing and topic formation for discourse (Breuker, 1990).

EUROHELP was partially funded by the ESPRIT Program of the European community under contract P280. The project encompassed an effort of about 112 person-years over a 5.5 year period. Partners in the project were: AXION (Denmark), ICL, University of Leeds (U.K.), Courseware Europe, University of Amsterdam (The Netherlands).

Acknowledgements: Work on the structure of the AM was distributed over all partners in the EUROHELP project and many people contributed of whom we would like to mention specifically: John van der Baaren, Cuno Duursma and Mick Smith, co-authors of earlier papers on this subject.

Notes

1. The correlation comes from the fact that operational knowledge, i.e., how to achieve states, is more easily expressed as procedures, while support knowledge, which has a strong structural bias, leads itself more easily to be described in a declarative way, than vice versa. It is confusing that the term declarative knowledge is often used both for the type of formalism as the type of knowledge itself.
2. Are they defined at the right conceptual level? It would also have been possible to define all system procedures in terms of the primitive 'assignment' action, but that obviously is too low level. If the set operation level turns out to be the right one for any IPS, this has consequences for education in Computer Science, e.g., programming languages like LISP are much closer to this level than for instance C or PASCAL.

References

Brachman, R.J. (1979). On the epistemological status of semantic networks. In: N.V. Findler (ed). Associative Networks. Academic Press, NY.

Breuker, J.A. (1988). Coaching in Help Systems. In: J. Self (ed). *Artificial Intelligence and Human Learning. Intelligent Computer-Aided Instruction*. London: Chapman & Hall.

Breuker, J.A. (ed) (1990). *EUROHELP: Developing Intelligent Help Systems*. EC, Copenhagen.

Breuker, J.A., Winkels, R.G.F. & Sandberg, J.A.C. (1987). A Shell for Intelligent Help Systems. *Proceedings of the 10th IJCAI, 1*, pp. 167-173.

Chandrasekaran, B. (1987). Towards a functional architecture for intelligence based on generic information processing tasks. *Proceedings of the 10th IJCAI, 2*, pp. 1183-1192.

Clancey, W.J. (1983). The epistemology of a rule based system—a framework for explanation. *Artificial Intelligence, 20*, pp. 215-251.

Davis, R. (1983). Reasoning from first principles in electronic trouble shooting. *International Journal of Man Machine Studies, 19*, pp. 403-423.

Duursma, C.M. (1988). Representing effects of commands in terms of changes in a structure of objects. Report COE-EUROHELP-025.

Genesereth, M.R. & Nilsson, N.J. (1987). *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, California.

Moran, T.P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies, 15*, pp. 3-50.

Newell, A. (1982). The Knowledge Level. *Artificial Intelligence*, pp. 87-127.

Sathi, A., Fox, M.S. & Greenberg, M. (1985). Representation of Activity Knowledge for Project Management. *IEEE Trans. Pat. Anal. & Mach. Intell., 7*, pp. 531-552.

Schank, R. (1975). *Conceptual information processing*. North Holland, Amsterdam.

Steels, L. (1987). The Deepening of Expert Systems. *AI Communications, 0*, pp. 9-16.

Wielinga, B.J. & Breuker, J.A. (1986). Models of Expertise. *Proceedings of ECAI'86*, pp. 306-318.

Winkels, R.G.F., Breuker, J.A. & Sandberg, J.A.C. (1988). Didactic Discourse in Intelligent Help Systems. *Proceedings of ITS-88*, Montreal, pp. 279-285.